VIENNA
UNIVERSITY OF
TECHNOLOGY
Institute of Chemical Engineering

Dissertation

Gasification in Power to Gas concepts

Simulation of advanced process configurations

written at TU Wien,

for the purpose of obtaining the academic degree

"Dr.techn." supervised by

Univ.Prof. Dipl.-Ing. Dr.techn. Hermann Hofbauer

E166, Institute of Chemical Engineering

approved by the

Faculty of Mechanical Engineering

Dipl.-Ing. Stefan Hemetsberger

0625929

Weinbergweg 31

4880 St. Georgen/Att.

January 12th, 2017

_____

Stefan Hemetsberger

Acknowledgement

This dissertation was created in the course of my work as research assistant at the Institute of Chemical Engineering at TU Wien.
I would like to thank everyone who has helped me, in any variety of ways, to realize this work.

First of all, I would like to thank my supervisor Univ. Prof. Dr. Hermann Hofbauer for the opportunity to work in the working group for Future Energy Technology. It has been a pleasure to do this thesis under his guidance. I would also like to thank him for his technical input as well as the freedoms he gave me for creating this work.

Further, I would like to thank Ass. Prof. Dr. Michael Harasek, for his input in general and in particular for the guidance through the FFG-project *Entwicklung eines katalytischen Prozeses zur Methanisierung von CO_2 aus industriellen Quellen*. Representative for all the team members of his working group I would like to thank Dr. Martin Miltner for the good cooperation during this project and the discussions we had.

I would like to thank all my colleagues, from our and other working groups, I spent time with during my time at the university. Thank you for any kind of assistance and the time we spent together during and outside the working time. Namely I would like to thank Dipl.-Ing. Florian Benedikt, Dipl.-Ing. Michael Kraussler, Dr. Stefan Kern and MSc. Jakob Luchner for reading the manuscript of this thesis and other assistance.

Last but not least, I would like to thank my family and friends who all of them helped me some how to do and finish this thesis. Especially, I would like to thank my parents who enabled me to study at all.

Kurzfassung

Aufgrund schädlicher Einflüsse auf unsere Umwelt und um eine Langzeit Energieversorgung zu sichern, ist es notwendig fossile und nukleare Energieträger ersetzen. Die Produktion erneuerbaren Stroms durch Wasserkraftwerke hat ihre maximale Kapazität bereits erreicht. Für die Bereitstellung der benötigten Energie sind Wind- und Solar-Energie vielversprechende Zusatzoptionen. Da die beiden letztgenannten Energiequellen nur unstetig verfügbar sind, ist es notwendig, Energiespeichersysteme in das Energienetz zu integrieren. Eine hervorragende Option ist die Speicherung von Energie in chemischer Form. Dabei wird Wasser durch überschüssige Elektrizität in Wasserstoff und Sauerstoff gespalten. Der Wasserstoff wird entweder für die Methanierung von Kohlendioxid und/oder Kohlenmonoxid verwendet, direkt in das Ergasnetz eingespeist oder in geleerte Erdgaslagerstätten gepumpt. Dieses Konzept ist unter dem Namen Power-to-Gas bekannt. Das Thema dieser Dissertation war es unter anderem zwei, verschiedene Power-to-Gas Konzepte mittels Prozesssimulationssoftware zu modellieren.

Für die Simulation der Prozessketten ist es notwendig, eine Modellbibliothek mit den verwendeten Komponenten zu haben. Über die Jahre hinweg wurde Knowhow auf dem Gebiet der Biomassevergasung, Methanierung sowie Gasaufbereitung und Reinigung in der Arbeitsgruppe Future Energy Technology (FET) gesammelt. Das Ziel war es das generierte Knowhow in einen PtG-Prozess, mit einem Zweibett Wirbelschicht Vergaser (DFB) als Kohlenstoffquelle, einfließen zu lassen und somit eine höchst mögliche Effizient zu erzielen. Ein zweiter Ansatz entstand im Rahmen eines Projekts mit anderen Forschungspartnern, finanziert durch die Österreichische Forschungsförderungsgesellschaft (FFG). Im Zuge dessen wurde eine Versuchsanlage zur CO_2 Methanierung, mit anschließender Produktgasreinigung durch eine Membran, erreichtet. Um eine genauere Vorhersage über den Prozessverlauf geben zu können, war es Teil des Projekts diesen zuerst in Form einer Prozesssimulation darzustellen.

Für beide Simulationen wurde die kommerzielle Simulationssoftware IPSEpro in Verbindung mit einer eigens in der FET-Arbeitsgruppe erzeugten Modellbibliothek verwendet. Um PtG Konzepte simulieren zu können war es notwendig die existierende Modellbibliothek um fehlende Komponenten, z.B. einem Elektrolyseur, zu erweitern. Neben der Prozesssimulation war es ein weiteres Ziel dieser Dissertation, den Inhalt aller Modelle der Modellbibliothek zu dokumentieren und die Richtigkeit deren Ergebnisse mittels einer Energie- und Massen-Bilanz zu belegen. Zusätzlich wurde die Software HSC 6.0 dazu verwendet, Ergebnisse von Gleichgewichtsreaktionen zu validieren.

Die PtG-Simulation mit Kohlenstoff durch Biomassevergasung wurde sehr detailliert durchgeführt. Je nach Bilanzgrenze errechnet sich ein Gesamtwirkungsgrad zwischen 58 und 85%.
Hingegen wurde für das PtG-Konzept mit Gasaufbereitung durch eine Membran nur eine sehr vereinfachte Simulation und Bilanzierung dieser durchgeführt. Keine Abwärmenutzung und auch keine kleineren Energie-Quellen und Senken fanden Berücksichtigung. Es er-

rechnet sich daraus ein Gesamtwirkungsgrad zwischen 55 und 67% für die Erzeugung von synthetischem Erdgas (SNG).
Zu guter Letzt ist neben der Effizienz des Prozesses auch noch die Qualität des erzeugten Erdgases von Bedeutung. Die errechneten Wirkungsgrade sind angemessen, verglichen mit anderen Studien. Für beide Prozesse konnte nachgewiesen werden, dass eine Einspeisung in das bestehende Erdgasnetz zulässig ist. Die Zulässigkeit der Einspeisung wurde dabei bestimmt durch den Wobbe Index.
Die Auswertung der Validierung und Bilanzierung der einzelnen Bibliotheksmodelle beweist für fast alle deren korrekte Funktion.

Abstract

Due to environmental aspects and reasons of long term supply, fossil and nuclear energy production have to be replaced. Sustainable electricity production through hydroelectric power has reached its limits. Other energy sources as wind and sun power are promising additions to cover the required supply. For a sustainable future of electricity production, efficient energy storage systems have to be established due to the fluctuating availability of wind and sun. An option is chemical storage, where electricity is converted into gas and stored within the pipeline grid or in exhausted caverns. This concept is known as Power-to-Gas (PtG). For this thesis, process simulation is applied to provide projections of two different Power-to-Gas concepts.

Throughout the years, knowledge in the fields of biomass gasification, methanation and gas treatment respectively purification has been gained in the working group Future Energy Technology (FET). The aim of the first PtG-concept was to combine this knowledge and to create a high efficient Power-to-Gas process with carbon supply through biomass gasification.
The second PtG-approach was created in terms of a project with other research partners, funded by the Austrian Research Promotion Agency (FFG). In the course of this project a test rig was created for CO_2-methanation and membrane purification of the synthetic natural gas (SNG). To have a proper forecast on the process, process simulation was applied.

For the modelling of different Power-to-Gas concepts the commercial simulation software IPSEpro and its model library BG_Lib are used. The model library itself was created by the FET working group. To be able to simulate PtG-concepts it was necessary to extend the model library by missing process units.
Besides the simulation of the PtG-projects, another objective of this work was the documentation of the BG_Lib model library. Not only the latest modifications of the BG_Lib library are documented, all the library's process units are described and validated through a mass and energy balance. For the BG_Lib documentation, every single process unit model was simulated in IPSEpro and the obtained energy and mass balances were exported to Matlab for their illustration. Additionally to IPSEpro, the software tool HSC Chemistry 6.0 was used to validate equilibrium reaction results of IPSEpro models.

For SNG production through biomass gasification, all energy sinks and sources are considered in the simulation. Depending on the system boundaries, an efficiency between 58 and 85% is calculated.
For the FFG-project's PtG-process its heat recovery is not considered as well as other minor energy sinks and sources. The efficiency results are between 55 and 67%.
In the end, the overall efficiencies of the processes and the gas quality of the created SNG are of interest. It has been shown that the produced synthetic natural gas meets the re-

quirements for an injection into the natural gas grid. The obtained efficiencies seem to be reasonable when comparing them to related work. For both Power to Gas concepts the requirements -e.g the Wobbe Index- for natural gas grid injection are met.
The documentation of the BG_Lib model library has proven that almost all process units are providing reasonable results and fulfil the mass and energy balance.

Contents

List of Figures

List of Tables

Nomenclature

Shortcuts

ACC	Activated char coal
AEL	Alkaline electrolyser
AER	Absorption enhanced reforming
BiP	Bipolar plates
BM	Biological methanation
CFB	Circulating fluidized bed
CGE	Cold gas efficiency
CHP	Combined heat and power plant
COGAS	Combined gas and steam
CTU	Conzepte Technik Umwelt AG
db	Dry base
DeNOx	Denitrification
DFB	Dual fluidized bed
DI	Deionized
ECN	Energy Research Centre of the Netherlands
EF	Entrained flow
EFC	Equilibrium floating conditions
FFG	Austrian Research Promotion Agency
FT	Fischer-Tropsch
GHSV	Gas hourly space velocity
GoBiGas	Gothenburg biomass gasification
HCR	Honey comb reactor
HTEL	High temperature electrolysis
IRSOFC	Internal reforming solid oxide fuel cell
KIT	Karlsruher Institut für Technologie
LHV	Lower heating value
LPM	Liquid phase methanation
MCFC	Molten carbonate fuel cell
MDK	Model development kit
MEA	Monoethanolamine
MEAS	Membrane electrode assembly
MUL	Montan Universität Leoben
ORC	Organic rankine cycle
PAFC	Phosphoric acid fuel cell
PCM	Phase change material
PEM	Proton exchange membrane
ppb	Parts per billion

ppm	Parts per million
PSA	Pressure swing adsorption
PSI	Paul Scherrer Institut
PtG	Power-to-Gas
RME	Rapeseed methyl ester
SER	Sorption enhanced reforming
SF	Stoichiometric factor
SNG	Synthetic natural gas
SOEL/C	Solid oxide electrolysis cell
SOFC	Solid oxide fuel cell
waf	Water and ash free
WGS	Water gas shift
ZSW	Zentrum für Sonnenenergie- und Wasserstoff- Forschung Baden-Württemberg

List of symbols

ΔH	Reaction enthalpy	$\frac{kJ}{mol}$
$\Delta h^0_{f,298}$	Specific enthalpy of formation at standard conditions	$\frac{kJ}{kg}$
δ	Difference to chemical equilibrium	-
$\dot{m}$	Mass flow	$\frac{kg}{h}$
η	Total plant efficiency	-
η_C	Carnot coefficient	-
$\eta_{isentropic}$	Isentropic efficiency	-
$\eta_{mechanic}$	Mechanic efficiency	-
λ	Air equivalence ratio	-
φ	Relative humidity	-
A_m	Membrane area	m^2
d	Relative density of gas and air	-
dt_{max}	Maximum temperature difference	K
dt_{min}	Minimum temperature difference	K
G	Gibbs enthalpy	kJ
H	Enthalpy	kJ
$h_{p,T}$	Specific enthalpy	$\frac{kJ}{kg}$
h_{p_0,T_0}	Specific enthalpy at standard conditions	$\frac{kJ}{kg}$
h_{total}	Specific total enthalpy	$\frac{kJ}{kg}$
HHV	Higher heating value	$\frac{kJ}{kg}$
J_i	Transmembrane flow	$\frac{Nm^3}{h}$
k	Heat transition coefficient	$\frac{W}{m \cdot K}$
K_p	Pressure related equilibrium constant	-
LHV	Lower heating value	$\frac{kJ}{kg}$
M	Molar mass	$\frac{g}{mol}$
m	Mass	kg
P	Electric power	kW
p	Total pressure	bar
P_i	Permeance of substance i	$\frac{Nm^3}{m^2 \cdot bar}$
p_i	Partial pressure of species i	bar

R	Ideal gas constant	$\frac{J}{mol \cdot K}$
S	Entropy	$\frac{J}{K}$
S_{ij}	Selectivity between species i and j	-
SF	Stoichiometric factor	-
T	Temperature	°C
V	Volume	m^3
W_O	Wobbe index	$\frac{kWh}{m^3}$
y_i	Mole fraction of species i	$\frac{mol}{mol}$

1 Introduction

1.1 Motivation

The reasons and the need to replace fossil energy sources is nowadays well known in our society. The limitation, the harmful effects to the environment and the difficult political situations in almost all oil producing countries are discussed in the media every week.
Alternatives have been identified and already applied. In order to provide sustainable energy and resources in the way we are used to consume them, it is necessary to create a well-considered network of interacting generation facilities.
Hydropower plants are very efficient for electricity production, but their capacity has reached its limits. Further promising renewable electricity production methods are wind- and photovoltaic- power plants. Also for the application of wind and sun no primary energy carrier has to be bought. The only costs are related to investment, operation and maintenance. One drawback is the shift in time between electricity production and consumption (Fig.1.1). Besides the troubles of having no electricity available when needed, too much electricity produced but not consumed ends in an unstable frequency and collapsing electricity grid. Different case scenarios for the generation of wind power in Austria were investigated within a master's thesis project [45], which was co-supervised as a part of this Phd-thesis.

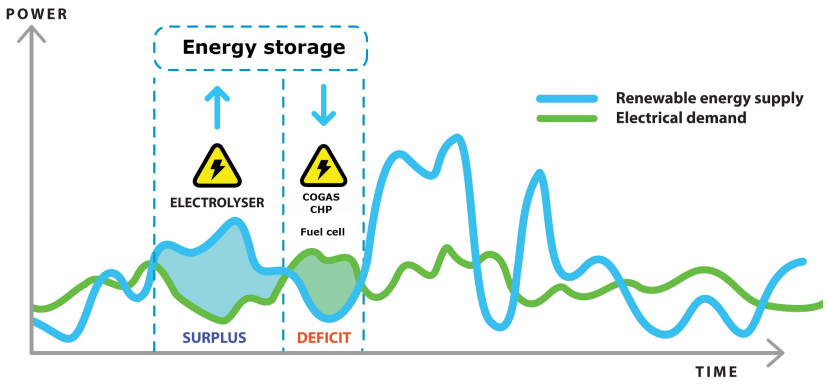


Figure 1.1: Fluctuating feed-in of wind- and solar-power [4][62]

Due to the fluctuating electricity production, load management and grid extension are required for better electricity distribution and storage.
Therefore, new and innovative ideas for energy storage have to be created and applied. One promising approach is the Power to Gas (PtG) concept. Water is split in an electrolyser into hydrogen and oxygen by excess electricity and further converted into synthetic natural gas (SNG) through the addition of carbon (e.g. CO, CO_2).

The natural gas grid is having the largest energy storage potential in the middle European area. Also research on an individual hydrogen grid and the addition of hydrogen to the natural gas grid is still on going. Hydrogen is more attractive for energy storage because no methanation unit and carbon source are necessary. [64] Further, in contrast to methane, hydrogen does not produce CO_2 when it is used, e.g. for combustion. However, exclusive hydrogen grids are currently just available in small scale e.g. in the German Ruhrgebiet and the addition of hydrogen to the natural gas grid is restricted by application of the gas mixtures.[35] For a H_2-content above 20%, in the natural gas grid, harmful effects could occur for gas turbines or if SNG is used as process gas or fuel. Additionally, the transport of gas mixtures of higher hydrogen contents ($\gg$20%) -in conventional natural gas grids- would require changes as for the measuring instruments and the maintenance efforts, but in general it is not a limiting factor.[52] So far the hydrogen content in the Austrian natural gas grid is restricted at 4 vol.% according to the guideline ÖVGW G 31.
Figure 1.1 shows a possible scenario for renewable energy production and electricity consumption. In times of electricity surplus, SNG and/or H_2 can be produced. In times of a deficit, electricity can be generated in a combined gas and steam power plant (COGAS) or in a combined heat and power (CHP) facility from SNG or a PEM fuel cell can use pure hydrogen to cover the electricity demand.

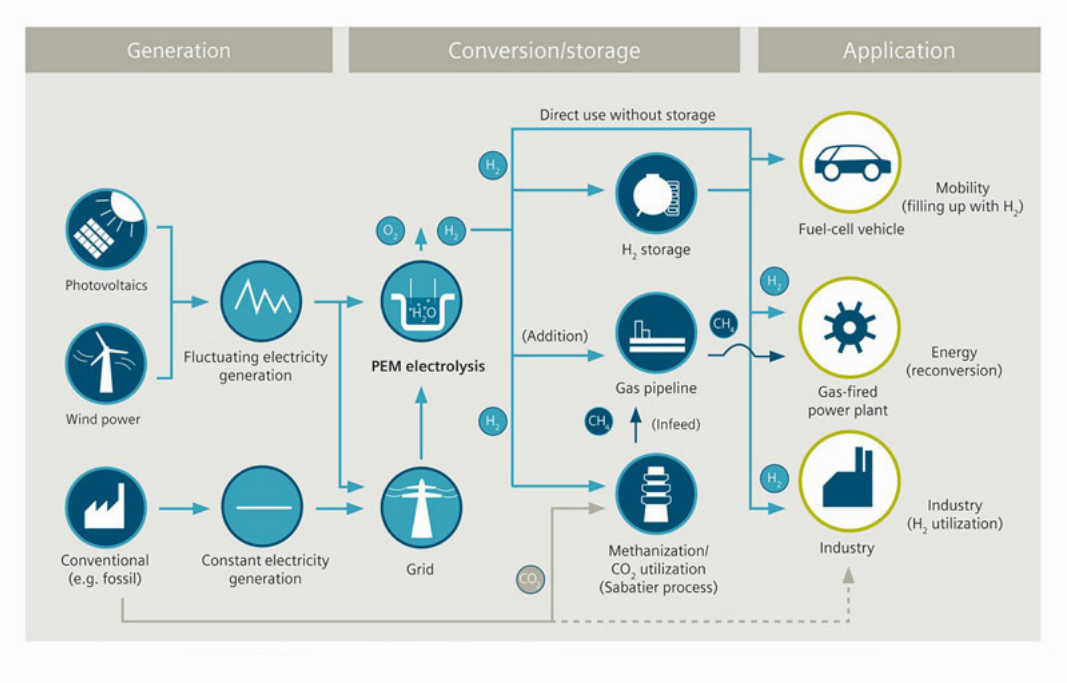


Figure 1.2: Scheme of Power to Gas utilization and the application of SNG and hydrogen [4]

Figure 1.2 provides an overview of the Power to Gas idea. The picture would have to be extended by the usage of CH_4 (SNG) for mobility and O_2 e.g. for industrial processes. For a completely renewable energy production, the conventional fossil plant -at the very left bottom- has to be replaced e.g. by a biomass gasification plant or any other technology with a renewable feedstock.

Instead of constant electricity generation through biomass, it is more interesting to produce renewable biofuels (SNG, FT-diesel,..) and biochemicals as there is no alternative to biomass as renewable carbon source for production in industrial scale (Figure 1.3). [57] This scenario is considered for the investigated Power to Gas concept of Section 3.1. A biomass gasifer produces syngas that is mixed with hydrogen from excess electricity conversion, to form SNG. In times of no excess electricity, the syngas can be applied for any biofuel or biochemical production. The concept is highly innovative as the desired products are gained in a production plant without any chimney, no emissions are released to the environment. All the created carbon dioxide and carbon monoxide is converted into SNG through methanation.

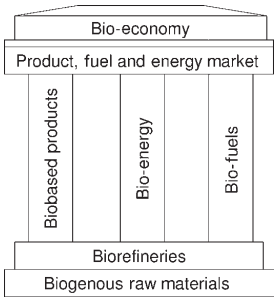


Figure 1.3: Pillar model of a future biobased economy [57]

As already mentioned, the application of biomass is more attractive for products where carbon is an indispensable prerequisite of the product, see Figure 1.4. In comparison to sun and wind, biomass can be stored and therefore fed whenever required. But biomass has to be harvested, transported and sometimes pre-treated as well. These efforts and costs are the main drawbacks besides a rising purchase price of biomass at lowering feed-in rates for renewable electricity.

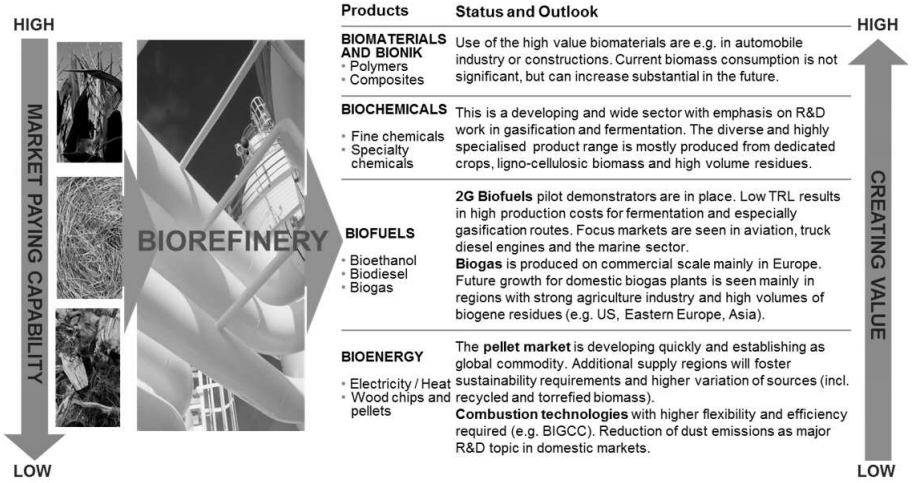


Figure 1.4: Future perspective for biomass application as primary energy carrier [85]

1.2 Related work

The topic of the listed publications are similar to this work's PtG concept of biomass gasification as carbon source. Some of the provided information was used as a starting point for the aims and process design right at the beginning of this dissertation and some was used throughout this work.

If using a biomass gasifier as carbon source for a SNG-related Power to Gas concept, it is desirable to provide syngas with a high methane content to the methanation reactor. Therefore, projects for SNG production through biomass gasification -without excess electricity hydrogen- are described as well (Subsection 1.2.1).
In Subsection 1.2.2, previous work on integration of biomass gasifier in Power to Gas concepts is discussed.

1.2.1 Research on biomethane production through biomass gasification

Synthetic natural gas can be produced via biomass gasification and an additional methanation step. The carbon conversion is limited due to a lack of hydrogen. However, the following three research institutes proof the feasibility of this concept.

TU Wien

In Güssing, Austria, bioSNG was produced by a 1 MW methanation unit. The product gas was provided as a side stream of a 8 MW biomass gasifier. For gasification the dual fluidized bed (DFB)-process was applied. The overall process chain of the power plant is presented in Figure 1.5.

The gas upgrading procedure of Figure 1.5 is described in detail in the dissertation of B. Rehling [81]. A first demonstration scale methanation unit (10 kW) was designed and prepared by the Paul Scherrer Institute (PSI), Switzerland. Parallel to the practical research, a simulation of the CHP plant Güssing and the bioSNG unit was created, based on experimental data. The overall efficiency from biomass to SNG was calculated as 66%. During the test runs in Güssing it was proven that the produced bioSNG was ready for pipeline injection.

Chalmers University of Technology

The research of Chalmers University was focusing on the creation of an optimized process chain for SNG-production and quantifying its cost- and CO_2-reduction potential. Heat and power production (CHP) are included in the efficiency considerations as well. This research work is closely linked to the Gothenburg biomass gasification project (GoBiGas). The project started in October 2007 and ended in August 2013.[73]

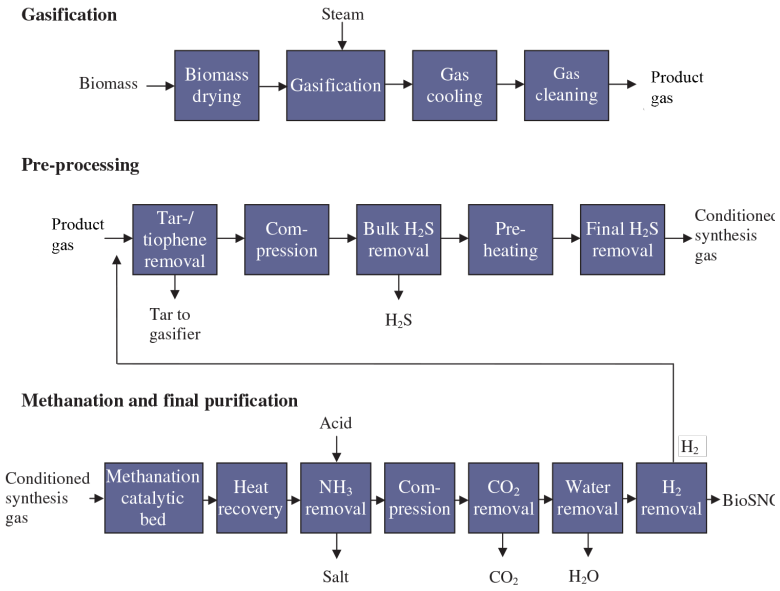


Figure 1.5: Schematic illustration of the Güssing BioSNG plant [82]

The overall idea was to combine power production with a steam cycle and SNG production. The process chain for SNG production was investigated at the beginning of the project in 2008. It is shown in Figure 1.6 and described in detail in [48]. At a first attempt, the overall efficiency for SNG production was about 60%.[48]
From the 100 MW gasifier, 26.4 MW of excess energy (char) are available. Additionally 2.8 MW of excess heat are served to a power steam cycle and 59.4 MW of SNG are produced (Figure 1.7).
The MEA regeneration is identified as the main energy consumer. Steam consumption of 5 MJ/kg CO_2 can even lead to a heat deficit.

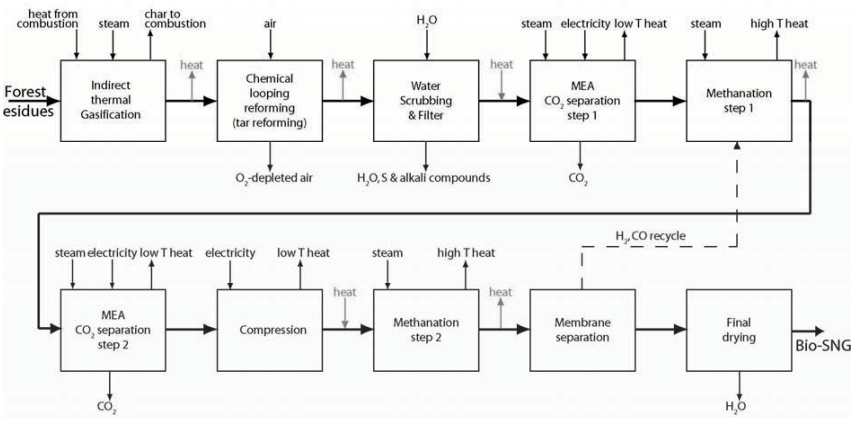


Figure 1.6: Proposed concept for the production of synthetic natural gas from biomass gasification [48]

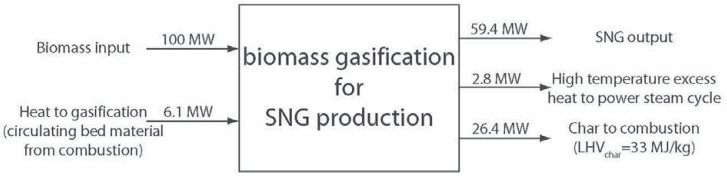


Figure 1.7: Heat energy balance for the base case SNG production process [48]

Further optimisation investigations on the choice of biomass drying, methanation system (adiabatic, isotherm) but also a general literature review on alternative options for every unit were done.[47][49] According to the project's latest publication, the overall efficiency between the LHV-based energy content of the SNG and the wet biomass is 69.4%.[49]

Energy Research Centre of the Netherlands - ECN

Also ECN does research on the topic of methanation. Figure 1.8 shows the process chain design for SNG production through biomass gasification. Based on this process chain, a

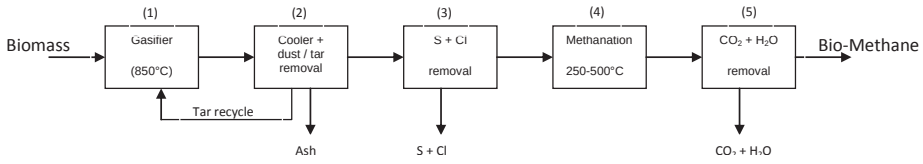


Figure 1.8: Simplified process scheme for production of Bio-Methane by gasification [106]

gross and net efficiency are calculated. For the gross efficiency neither the produced nor the consumed amount of electricity are considered. For the net efficiency it is assumed that the SNG is used for electricity generation with an electrical conversion efficiency of 60%. Excess process heat is assumed to be converted into electricity by a turbine.

Figure 1.9 shows the efficiencies for different gasifier technologies. EF stand for an entrained flow reactor, the circulating fludized bed gasification CFB is fluidized by oxygen. Allothermal gasification means indirect gasification, the same principle of a separated gasification and combustion reactor as it will be presented for the DFB and SER technology in Subsection 2.3.2. The gross and net efficiencies are highest for indirect gasification. This is caused through low losses of unconverted carbon and heat.
The gasifier's product gas contains a high percentage of hydrocarbons. According to [32] these hydrocarbons are converted into SNG at a higher efficiency than conventional gasifier product gas. More information on all the presented efficiencies of Figure 1.9 can be found in [32].
A detailed description of an 500 hour bio-SNG plant test run of the is described in [29].

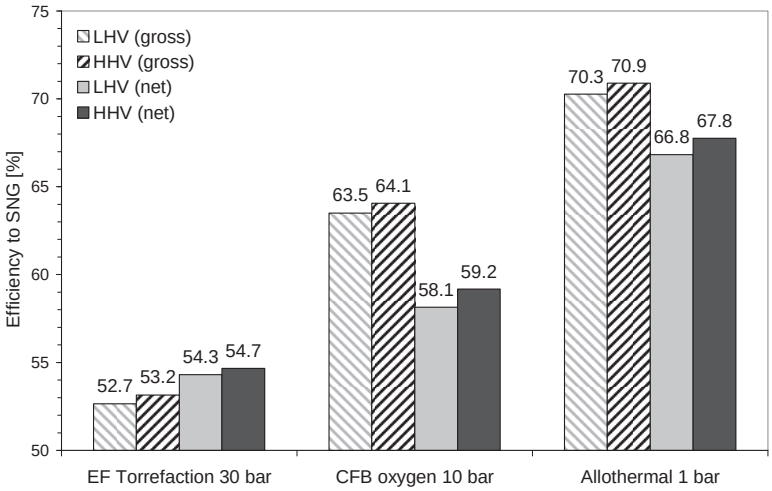


Figure 1.9: Gross and net efficiency of the various gasifier concepts either LHV or HHV based [32]

Zentrum für Sonnenenergie- und Wasserstoff- Forschung Baden-Württemberg - ZSW

In the course of AER-gasifier application, investigations on SNG production were done and a simulation for ”Biomass to Gas” was created.

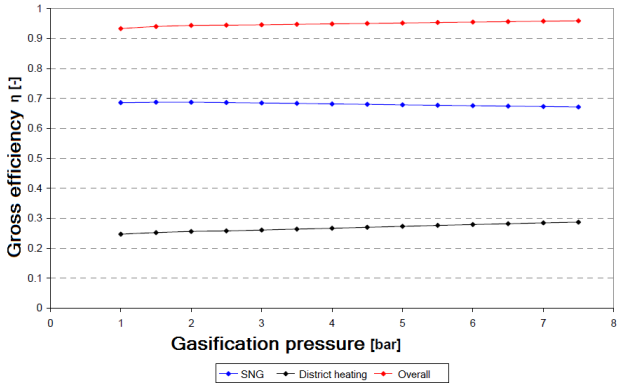


Figure 1.10: Gross efficiency in dependency of gasification pressure [38]

The gross efficiency (ratio between chemical energy of SNG and biomass) of the considered process chain is presented in Figure 1.10 in dependency of the pressure inside the gasifier. [38]

1.2.2 Biomethane through biomass gasification in a PtG-concept

The basic idea of the following projects is similar to the concept of Section 3.1. Namely, the use of gasified biomass as carbon deliverer for the Power to Gas approach. But every

process chain is different in its technical design and all the projects have a different focus. The ECN study has the objective to generate SNG with a very low hydrogen content through an enhanced methanation procedure, while ZSW spends a lot of efforts on diverse sensitivity analysis of the gasifier operation. Also the methanation systems, respectively their design is different for each study.

Energy Research Centre of the Netherlands - ECN

In contrast to the conventional SNG production, as described in Subsection 1.2.1, the SNG amount can be doubled through hydrogen addition.[87] It is no longer necessary to remove CO_2 before the methanation reactor if additional hydrogen is available through excess electricity from an electrolyser unit. All the CO and CO_2 can be converted. Due to strict hydrogen limitations of 0.5 vol.% in the Netherlands, a special sorption enhanced methanation process -in a three step inter cooled fixed bed methanation section (Lurgi HT)- was introduced as conventional methanation methods can not provide this low hydrogen level for their product gas. According to [70] a conversion efficiency of CO and CO_2 of almost 100% can be achieved at an overall efficiency of 70% similar to the standalone plant of Subsection 1.2.1.
However, also the sorption enhanced methanation hardly reaches the desired H_2-level of 0.5 vol.%.[87]

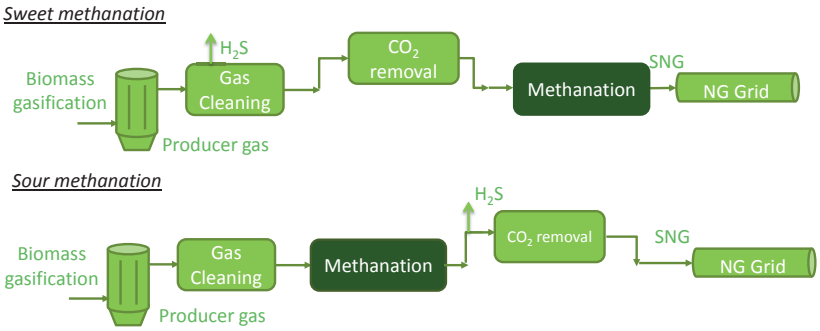


Figure 1.11: Process scheme of sweet and sour methanation [88]

In contrast to the work of Van der Meijen [32], it is distinguished between sour and sweet methanation, see Figure 1.11. In the sweet case H_2S and CO_2 are removed before the methanation step. For the sour case sulphur is removed downstream of the methanation unit, just NH_3 and HCl are removed before. CO_2 is removed if no excess hydrogen is available. For the sweet methanation, monoethanolamine (MEA) is applied as scrubbing agent and for the sour case the sulfinol process is used as it is capable to handle large changes in the acid gas concentration which are expected when switching between the operation with additional hydrogen and without. Further information on sour methanation as well as sorption enhanced methanation can be found in [70].
There is no significant difference of the process efficiencies for sweet and sour methanation, but a simplification of the process chain for sour methanation as an additional pre reforming step is necessary for sweet methanation. For the sour case, the reforming appears in the methanation reactor due to a different catalyst choice. [70]

Zentrum für Sonnenenergie- und Wasserstoff- Forschung Baden-Württemberg - ZSW

In the PhD-thesis of J. Brellochs [23], the production of SNG through AER-gasification of biomass was coupled with electrolyser-hydrogen. Also the case of no excess electricity is investigated. Oxygen from the electrolyser is used for OxyFuel combustion in the gasifier's combustion chamber. If oxygen is avialable, flue gas (mainly CO_2) is also converted into SNG but in a separate reactor. The reactor types have not been specificly defined. Oxygen of the flue gas is assumed to react inside the flue gas methanation reactor with hydrogen to water. Therefore, it is not considered as a potentially harmful component for the methanation catalyst. For product and flue gas upgrading see [23]. As electrolysis technology, pressurized electrolysis at 10 bar has been proposed. Excess process heat is converted into electric power via an Organic Rankine Cycle (ORC) process at an electric efficiency of 17%.[23] Based on [23], an overall efficiency of 70-75% for excess electricity supply is estimated.

1.3 Aim of the work

This thesis has two principal objectives.
Different Power to Gas scenarios should be simulated with the software tool IPSEpro.

At first, biomass is gasified for carbon supply to the methanation reactor. The simulation and its results are documented in Section 3.1. The basic concept is to continuously operate a dual fluidized bed biomass gasifier for syngas production and to use both -the gasifier's syngas and flue gas- as carbon source for SNG generation if excess electricity is available. This dissertation applies science-of-the-art knowledge to set up a process chain optimized for SNG production.

The second Power to Gas approach, assumes carbon dioxide from biogas production for the methanation process. This simulation was created to give a forecast on operation conditions for an actual laboratory scale methanation test facility with a membrane unit as final purification step, see Section 3.2.

As an additional task, the purpose of this dissertation was also to create a documentation of the applied model library BG_Lib. For more than one decade students and employees, from the research group of Future Energy Technology, are extending and adjusting the library-file so that it was desirable not only to document the changes in terms of this dissertation, but also to document the full library source code, see Chapter 4.

Within the Fundamentals-Chapter 2, background information for the applied operation units is presented.
Information is also given on the fields of hydrogen- and thermal energy- storage. Both topics are of interest for Power to Gas solutions as it is desirable to continuously operate the methanation reactor, also in times of no excess electricity and to store waste heat as the PtG-process is strongly exothermic, especially if SNG is created.

2 Fundamentals

This chapter provides a description of the most essential process units of the simulated Power to Gas concepts presented in Chapter 3. Background information is given to the various process components.

The two main components of any power to gas concept are electrolyser and methanation unit. Therefore, they are described more detailed than any other unit. For the concept of SNG production through biomass gasification (Section 3.1), the DFB gasifier is as important as the electrolyser and methanation reactor. So it is considered as a third main component for this fundamentals description.

Other process steps as biomass drying or gas cleaning are more or less state of the art technologies and therefore not described that detailed. In Section 2.4 (Gas cleaning) only purification options are described which are actually used for removal of tar, sulphur and other impurities in the simulations.

Section 2.6 is more or less a literature review on storage possibilities of thermal energy. This is relevant as the Power to Gas process for SNG production is exothermic. Dependent on the desired waste heat application it is of interest to include long or short term heat storage systems into a Power to Gas process. In this context district heating/cooling is an interesting waste heat recovery.

Section 2.7 (Hydrogen storage) is included to give an impression of different storage opportunities as the availability of hydrogen in times of no excess electricity is one of the challenging topics for the Power to Gas idea, as it is desirable to continuously operate the methanation reactor.

2.1 Biomass dryer

A biomass gasifier unit becomes more efficient if biomass has the right moisture content. Due to drying, the biomass' lower heating value (LHV) is increased, see Figure 2.1. According to [54], a reduction of the biomass' water content from 35% to 20% results in fuel savings of 17% for the DFB gasifier. Whereas a water content below 20 wt% causes an increasing tar content in the gasifier product gas. Tar is responsible for filter blocking and fouling of heat exchanger.[54]

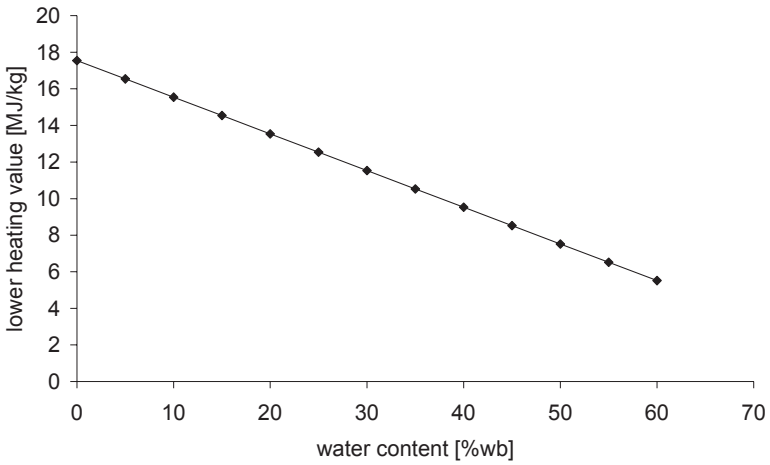


Figure 2.1: Lower heating value of beech wood in dependency of water content [54]

The highest energetic benefit is achieved if waste heat at a low temperature level is applied for drying, as it is hard to find any other utilization for low temperature waste heat.

Biomass drying can be done through conveyor-, drum, silo- or cross flow shaft-dryers. The last option has been chosen for the CHP plant Oberwart, in form of a moving bed dryer. Due to that, only this dryer technology is described in the following. The other drying options are presented in the work of [68] and [84].
Its technical design as well as the Oberwart power plant's efficiency increase through low temperature drying are shown in the dissertation J. Kotik [60].

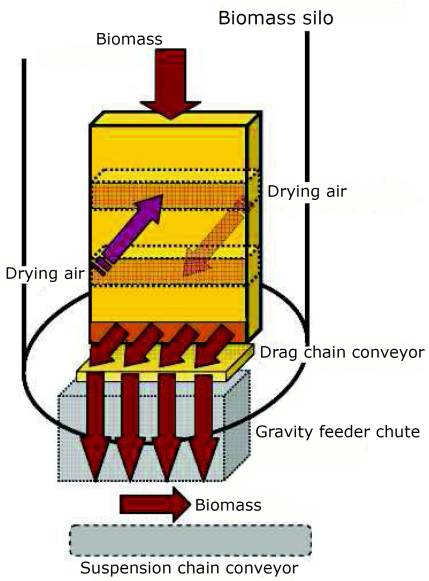


Figure 2.2: Scheme of Oberwart's moving bed dryer [84]

Figure 2.2 pictures a moving bed dryer within the biomass silo, as it is operated in Oberwart. Biomass is fed from the top, drying air enters from the side and in the end the biomass

ends up on the suspension chain conveyor by gravity. The most influential parameters on biomass drying behavior are listed in Table 2.1.

Parameter	Range
Temperature	48-72°C
Drying air velocity	0.2-18 m/s
Biomass particle thickness	11-168 mm

Table 2.1: Influencial parameters for biomass drying [68]

According to [68], the most influential parameter is the biomass thickness while other parameters as temperature and air velocity do not have significant influence as long as they are within the predefined range of Table 2.1.

2.2 Electrolysis

For the storage or use of regenerative excess electricity, process water is split by an electrolyser into two parts of hydrogen and one part of oxygen. Besides the process water, also cooling water is fed to the electrolyser and heated. This general principle can be seen in all the visualizations of the different electrolysis cell types of Subsection 2.2.1.

Representative operation conditions for the described electrolysis technologies can be found in Table 2.2

2.2.1 Electrolysis-cells

Alkaline electrolysis (AEL)

This is an industrially proven electrolysis technology. Its main application is the electricity conversion for overdimensioned power plants. The worlds largest AEL-electrolyser (156 MW nominal power) is operated at the Assuan-dam, Egypt. Its hydrogen output is 33000 m^3/h. The technology shows improvement potential for dynamic operation conditions which is relevant for renewable energy sources.

Figure 2.3 shows the basic structure of an AEL electrolyser. An ion transporting membrane (3) separates the cell into two half-cells, cathode (1) and anode (2) side. The membrane is embedded into the end-frame (5). Both electrodes are numbered as 4. They are connected to end-plates (7), which separate the system from the environment in case of just one single cell. For more than one cell, all the single cells are separated through these plates. In both half-cells water plus potassium hydroxide (KOH) lye (20-40 wt.% KOH) is circulating. Potassium hydroxide is added to increase the water's electrical conductivity. The lye is stored in tanks (8), besides lye storage the tanks are also responsible for gas-liquid separation. The electric energy source is marked by the number 6.

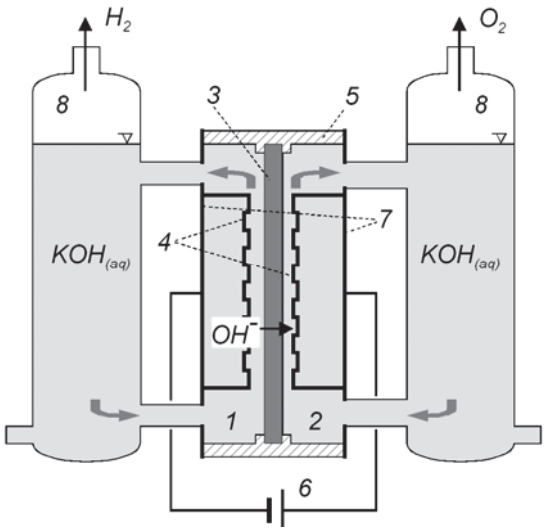


Figure 2.3: Schematic illustration of an alkaline electrolysis cell [96]

Due to an decomposition-voltage of 1.23 V on the electrodes, water is split at the cathode side (Eq.2.1), assuming ideal conditions.

$$2H_2O_l + 2e^- \rightarrow H_{2g} + 2OH^- \tag{2.1}$$

H_2 is formed and removed as product gas while the hydroxide ions (OH^-) are transported through the membrane. At the anode, two electrons are released from the hydroxide ion so that atomic oxygen and water are formed (Eq.2.2).

$$2OH^- \rightarrow \frac{1}{2}O_{2g} + H_2O_l + 2e^- \tag{2.2}$$

The two electrons are transported by the closed electric circle where the electric energy is fed as well. The overall reaction of both half-cells is applicable for any other cell system, see Equation 2.3.[101]

$$H_2O_l \rightarrow H_{2g} + \frac{1}{2}O_{2g} \tag{2.3}$$

Membrane electrolysis (PEM, PEMEL)

The short-form PEM stands for proton exchange membrane. Its technology is based on the developments for the proton exchange fuel cell. The ability for dynamic operation behavior and to run at pressurized conditions are beneficial for the application for renewable energy sources.

Both electrodes and the membrane are connected to a so called membrane electrode assembly (MEAS). On both sides of the MEAS current distributors (usually a solid polymer electrolyte) are installed. They are enabling access to the electrodes for current and water

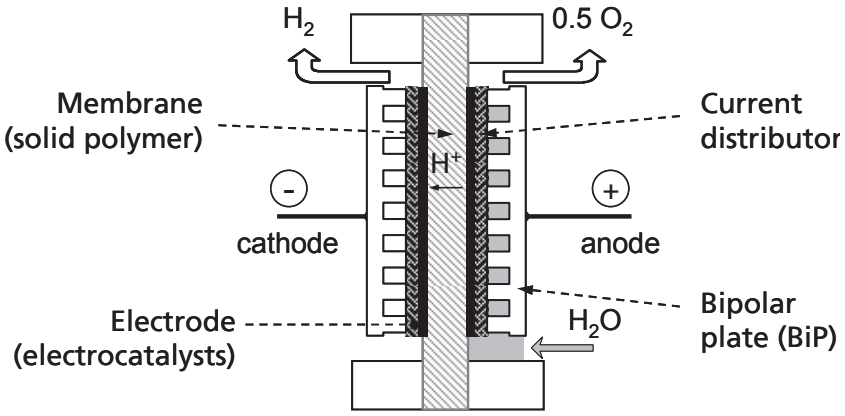


Figure 2.4: Schematic illustration of a PEM electrolysis cell [96]

and they are responsible for the removal of the product gases. Bipolar plates (BiP) are surrounding both half cells. The recesses in the BiP are called flowfield and installed to enhance water transportation to the electrode and product gas removal.
Contrary to the AEL type, water is just fed at the anode side (Fig. 2.4). There, it is split due the decomposition voltage, see Equation 2.4.

$$H_2O_l \rightarrow \frac{1}{2}O_{2g} + 2H^+ + 2e^- \tag{2.4}$$

The oxygen is forming O_2 molecules while the H^+ protons are transported through the membrane. On the cathode side hydrogen is formed with two additional electrons (Eq.2.5).

$$2H^+ + 2e^- \rightarrow H_{2g} \tag{2.5}$$

In theory, water is just present at the anode side and pure hydrogen is formed on the cathode side. The actual hydrogen product stream contains water as well. However, the content of unwanted components in the hydrogen stream is much lower for the PEM technology, compared to AEL. [96] [101]

High temperature electrolysis (HTEL, SOEL)

For this system, a share of the required decomposition energy of water is provided through heat at a temperature level of 850-1000°C. Therefore, the cell voltage can be reduced up to 25% to about 1 V. Due to that, higher -electricity based- efficiencies are obtained.
As presented in Figure 2.5, both half cells are separated by an oxygen conductive, solid electrolyte. At its outer boundaries the electrodes are fixed. On the cathode side super heated steam is added. It is separated -according to Equation 2.6- into hydrogen and oxygen-ions.

$$H_2O_g + 2e^- \rightarrow H_{2g} + O^{2-} \tag{2.6}$$

The oxygen-ions diffuse through the electrolyte and react at the anode side, according to Equation 2.7. [96] [101]

$$O^{2-} \rightarrow 2e^- + \frac{1}{2}O^{2-} \tag{2.7}$$

Each electrolysis-cell can be operated at atmospheric or pressurized conditions.

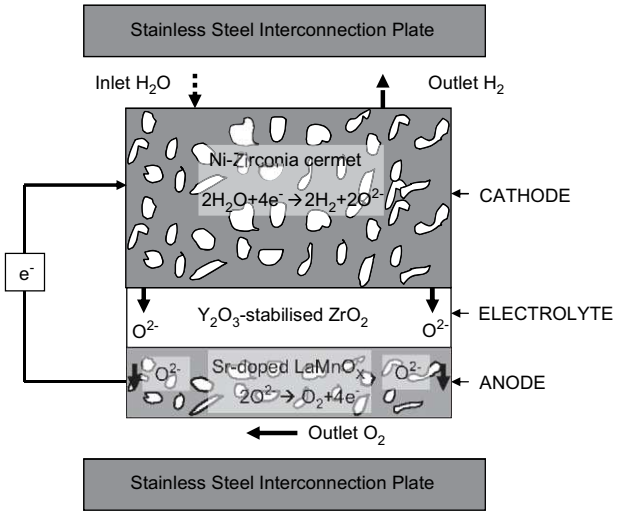


Figure 2.5: Schematic illustration of the HTEL electrolysis cell [24]

Atmospheric electrolysis (0.02-0.05 bar above atmosphere) is a stable and simple solution. The process design and system management are relatively simple. Its operation window starts from below 20% of full load to full load. A lot of operation experience is available and its investment costs are significantly lower in comparison to pressurized electrolysis. The relative amount of current processed per stack capacity (current density) is low which leads to higher space requirements. Product gas drying and compression are additional components to be implemented. [101]

Pressurized electrolysis units are commonly operated at around 30 bar. The tendency is to increase the pressure level. Test-runs are already made at 120 bar and higher. Due to the pressure increase possible pre-compression stages for further storage or other industrial use of hydrogen can be avoided. Also the size of the electrolyser and the related share of investment-costs will decrease for the higher pressure level. While the overall investment costs are higher for the pressurized model types, so are the maintenance and safety efforts due to the higher pressure. A further drawback is the limitation of the operation window to 30-100% of full load for pressurized operation above 10 bar. [101]

2.2.2 Stackdesign

The gas production rate of a single electrolysis-cell is limited. Therefore, for large amounts of gas production various cells are coupled to a so called stack. The cell connection can either be done monopolar or bipolar.

Monopolar stackdesign

The parallel cell connection ensures low system voltage for high current flux. A surplus of the parallel connection is, even if one cell turns down, the residual cells are still running

properly.

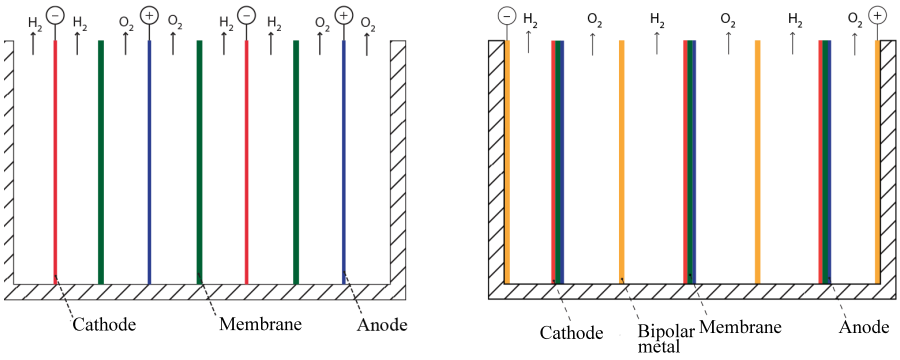


Figure 2.6: Scheme of monopolar (l.) and bipolar (r.) stack design [101]

Bipolar stackdesign

This solution is applied in order to increase the stack's system voltage and therefore decrease the inlet-current flux. Bipolar plates (BiP) are installed to separate the single cells. On the one side of the BiP is a cathode on the other an anode. Throughout this system, the single cells are straight connected in series. The drawback of this system is its dependency on every single cell. None of them can turn down without shutting down the whole stack. [101]

2.2.3 The electrolyser and its periphery

A complete electrolyser unit does not just consist of its stack. A lot of other components have to be implemented to provide the desired gaseous products. This subsection gives a short impression of the auxiliary parts. Basically, the difference between the periphery of an alkaline and proton exchange membrane is the absence of an electrolytic circle and lyes processing for PEM-electrolysis, see Figure 2.7.
In this figure one can also see, the stack is defined as the electrolyser's heart. Below the overarching parts, the system is separated into an up- and downstream section.

Overarching components

They are installed due to control and safety reasons. Gas transporting parts are flushed by nitrogen to minimize the risk of explosions. The supplied electric energy is mainly applied for electrolysis. On a yearly average less than 10% are used for auxiliary systems as valves. Ambient air control and product gas analysis are installed to discover timely changes and to react to them. Heat- and cooling-management is necessary to regulate energy interaction between electrolyte circle, gas cooling, condensation, gas-cleaning and compression.

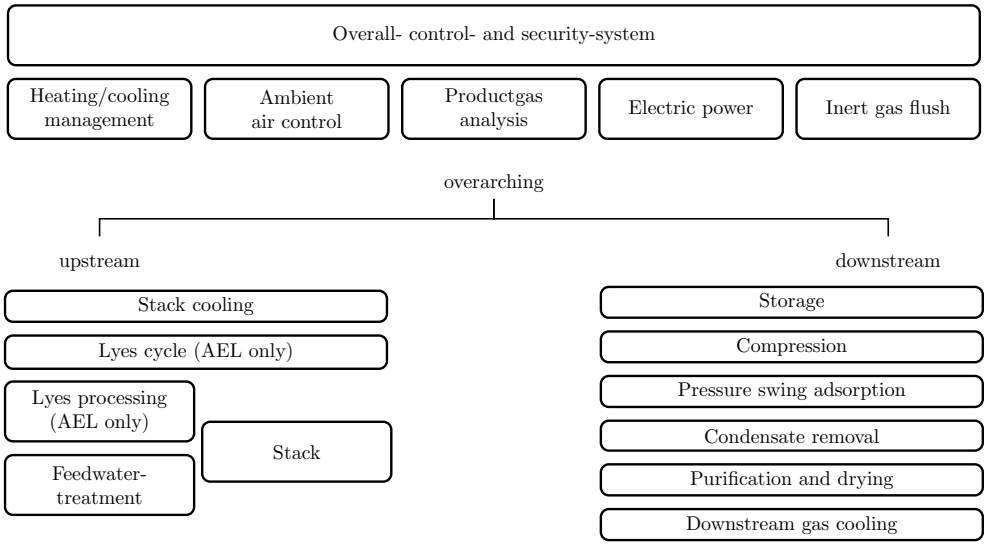


Figure 2.7: Electrolyser periphery [101]

Upstream

Before entering the stack, feed water is processed e.g. via reverse osmosis. Ions and suspended solids are removed. For AEL, lyes is circulating as well. It has to be processed and is floating through the stack (see Subsection 2.2.1), in addition to the electron transport it is also responsible for cooling the stack.

Downstream

After the stack, the product gas is cooled down to condensate remaining electrolytes. Afterwards, they are sent back to the electrolyte circle. The condensate is removed to protect downstream components from aggressive condensate. To gain a hydrogen purity of 99.9 vol.% it is purified and dryed as well. The residual water content can further be lowered through pressure swing adsorption (PSA) up to 99.999 vol.% H_2. Compression is reducing the hydrogen volume for storage. [101]

2.2.4 Comparison of electrolyser-technologies

Table 2.2 shows characteristic values for all of the three relevant electrolysis technologies. In comparison to [101], additional data has been added for the currently largest available PEM-electrolyser. The actual maximum capacity for PEM-electrolysis is about 6 MW and is going to be installed for the Energie Park Mainz [3]. Till 2018, Siemens is planning to build the third generation of their PEM-electrolyser series with an electric input capacity of up to 100 MW [26]. In [101] further information on the technical design and relevant calculations for electrolysis is given.

	AEL	PEMEL	HTEL
Electrolyte	Liquid - KOH	Solid - Membrane	Solid - Ceramic
T in °C	40-90	20-100	700-1000
p in bar	1-3	30-50	about 30
Efficiency in %	62-82	67-82	68-82
Stackdesign			
Structure	Bipolar/filter press	Filter press	Commercially not available
Active cell area	0.1-4 m^2	10-750 cm^2	up to 100 cm^2
Current density in A/cm²	0.2-0.45	up to 2.5	0.3-3.0
Cell voltage in V	<2.4	<2.2	
Cells per stack	up to 700	<120	
System design	Lyes circle, El. power, Gas separation/scrubbing, Compression, Fine cleaning	Similar to AEL but simpler system design, pressurized design	
Hydrogen production rate in m^3/h	1-760	0.01-30	5.7 for 18 kW unit
Electrical power	5 kW-3.4 MW	0.5-160 kW*	18 kW largest unit available
Lifetime incl. overhaul in yrs	20-30	10-20	no information
Energy consumption in kWh_{el}/m^3H_2	4.5-7.0	4.5-7.5	0.6-3.2
Part load operation in %	20-100	0-100	no information
Power density in W/cm^3	till 1.0	till 4.4	no information
Investment costs I_0 in EUR/kW, values from 2014	800-1500	2000-6000	no information
Construction and delivery costs	10% I_0	10% I_0	no information
Maintenance, operation and insurance costs	4% I_0 per annum	4% I_0 per annum	no information
Advantages	Cheap in construction and order, state of the art technology	Simple design as no electrolyte circle is present, smaller overvoltage due to noble metal catalysts	Cell voltage smaller for steam than liquid water, if high temperature energy is available (e.g. through chemical synthesis) it can be used for electrolysis
Disadvantages	Expensive lyes recovery, not made for dynamic operation	Higher material requirements and therefore more expensive materials	Separation of hydrogen and steam -at higher temperatures- are possibly causing problems
R&D tasks [95]	Increase of current density, faster dynamics of the complete system, higher part load range, decreasing production costs through economies of scale	Increase life time of materials/stack, scale up for stack and system, decrease of costs by reduction or substitution of expensive materials	Development of adoptable electrodes/electrolyte for SOEL, cell and stack design, proof of life time, pressure tightness, cycling stability

*latest figures: 1 MW[22] 5 MW[3], 2018:100 MW[26]

Table 2.2: Comparison of electrolyser systems [101]

2.2.5 Electrolysis applied for PtG solutions

Figure 2.8 pictures different manufacters and R&D-institutions as well as the status of development of their products. This figure was created by [99], they are using the short-cut AEC instead of AEL and SOEC/FC instead of HTEL. However, the same electrolyser technologies are meant. Information from the previous subsection is manifested by Figure

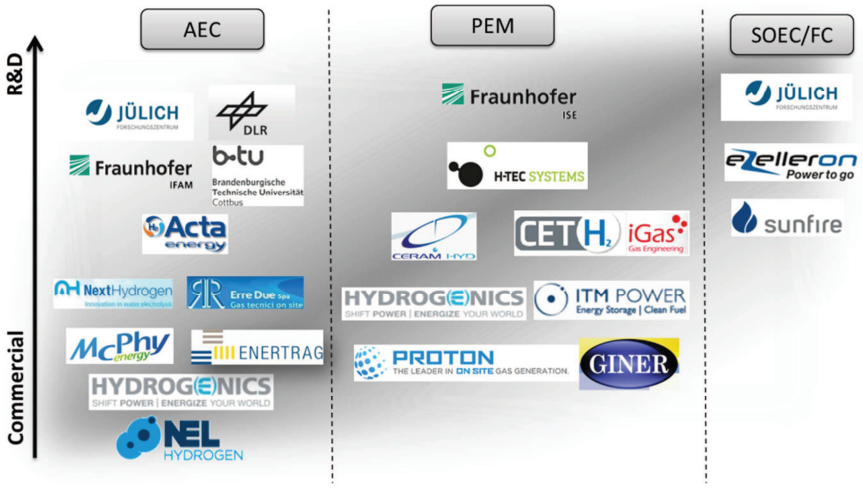


Figure 2.8: List of various electrolyser manufacturer [99]

2.8, saying that **AEL (AEC)**-technology is well established for commercial applications. Its largest manufacturers are NLT(Statoil), Hydrogenics and McPhy Energy. Plenty of research is done to fit the dynamic PtG-operation requirements. By the takeover of the electrolyser division of Enertrag, McPhy has become the largest AEL-manufacturer for 2-7 MW solutions, next to Hydrogenics.
PEM-electrolysis has started to catch up with the AEL technology. Stacks with a full load performance of 1 MW are already available. Well known for their PEM products are Proton, Giner, H-tec, CETH, ITM and Hydrogenics. Power-to-gas projects with an electrolyser performance smaller 1 MW have already been realized.
The **HTES (SOEL)** concept is currently in R&D status. Sunfire has announced market launch of an 200 kW HTES-electrolyser. Further potential manufacturer are assuming their products to be commercially available in 2020. A very detailed list on commercially applied electrolysers from all over the world and their regarding system data can be found in [99].

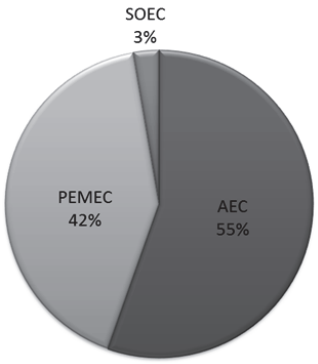


Figure 2.9: Applied electrolysis technologies for all PtG projects since 2003 [99]

For more than 50% of all PtG projects -throughout the last decade- AEL-electrolysis has been used. This equals about 70% of the installed overall power (Figure 2.9).
Most of these projects are hosted in Germany and Northern America. A list of international PtG projects is provided by [99]. Figure 2.10 shows an overview of the latest MW-scale projects in Germany and the applied electrolyser types. Research projects in Austria are listed in detail in [99] as well, but they are still operated in a low kW-scale.

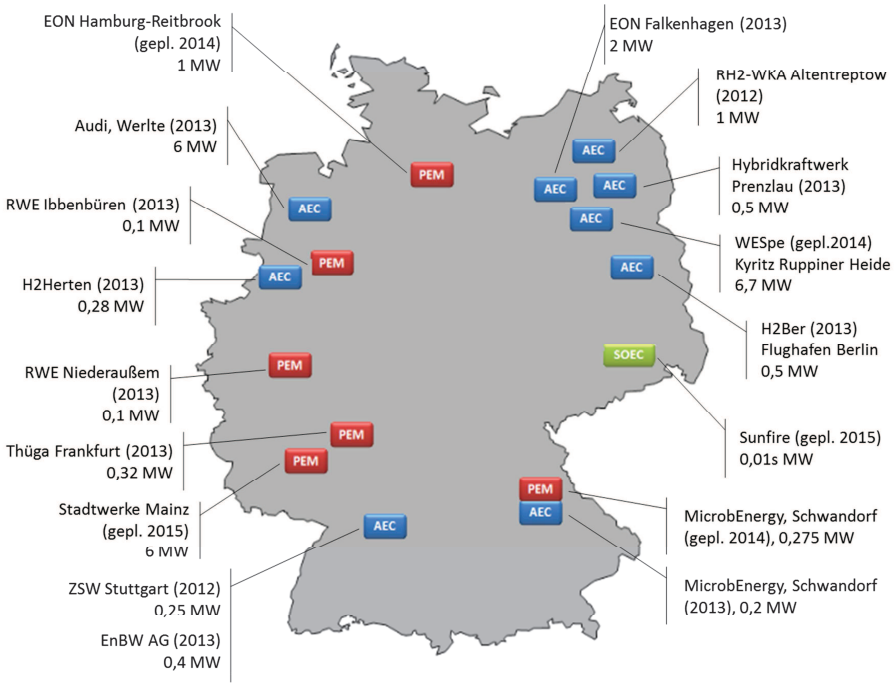


Figure 2.10: Overview of PtG projects in Germany [99]

2.3 Gasification

Gasification is a promising way to get sustainable valuable products from lignocellulosic biomass. A brief explanation of biomass gasification and the relevant reactor concepts is provided in this chapter.

2.3.1 Thermochemical conversion of biomass while gasification

Figure 2.11 shows that the different phases of thermochemical conversion can either be independent from each other, occure parallel or can just effect each other.[56] These phases can be distinguished by the excess oxidant ratio λ. The definition of λ and the different conversion phases are described in the following.

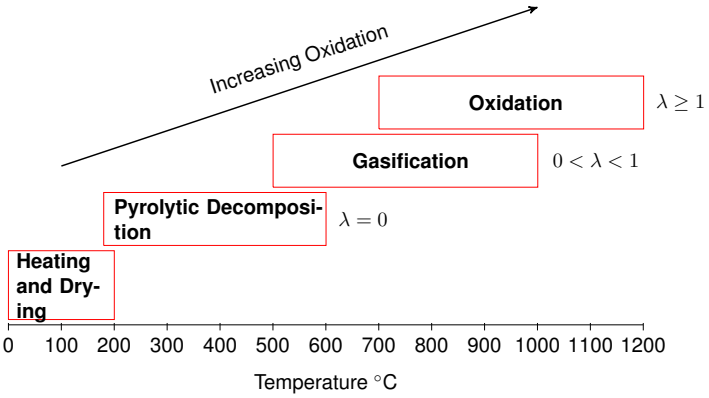


Figure 2.11: Phases of thermochemical conversion [56]

Air equivalence ratio λ is described by Equation 2.8.

$$\lambda = \frac{\dot{m}_{Oxidant,actually}}{\dot{m}_{Oxidant,stoichiometric}} \tag{2.8}$$

The air equivalent ratio is also known as air excess ratio. Fuel oxidation requires oxygen, which is usually delivered by air but can also be delivered through other oxygen carriers. In case of steam, the gasifier product gas does almost not contain any N_2. The actually delivered amount of oxygen for a combustion process is more than the stoichiometry of a reaction would require. Typical excess air ratios are [56]:

- Pyrolytic conversion $\lambda = 0$
- Gasification $0 < \lambda < 1$
- Combustion $\lambda \geq 1$
  - Stoichiometric combustion $\lambda = 1$
  - usual λ values for wood firing $1,5 < \lambda < 2,5$

Physical and chemical reactions for the single conversion steps of Figure 2.12 are described in the following sub-items.

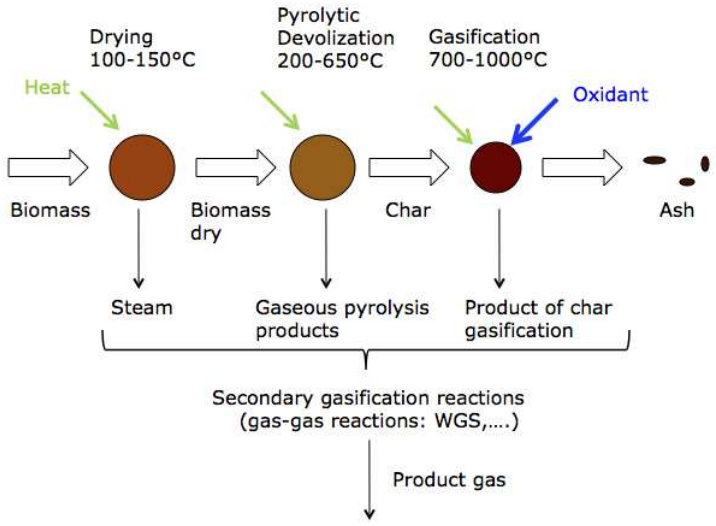


Figure 2.12: Scheme of a gasification process [56]

- **Heating and drying**
  This is an endothermic process to remove bounded water within the organic structures and porous spaces of biomass. The required duration for this process can decrease if cracks are built within the biomass structure because of different coefficients of expansion for varying directions or just because canals for steam transport are closed by melted resins. During this process, biomass temperature does not increase until almost all water is removed. [56]

- **Pyrolytic conversion**
  It describes cracking of macro molecules -which are in fact the basis of biomass- without the presence of oxygen ($\lambda = 0$), just by heat. 80-90% of biomass' organic components are converted into gas, the residuals mainly consist of carbon and ash. No interactions are detected between the various individual wood components according to Kaltschmitt et.al.[56]. If the desired product should be in gaseous phase, the heat up velocity of biomass should be high and a long residence time at a high temperature level (T >800°C) is desired.
  During gasification, long-chain or poly-aromatic hydrocarbons (tars) are cracked into shorter hydrocarbons like methane. This scenario is shown through the reactions represented by the reaction rate constants k_2 and k_4, in Figure 2.13. Rate constant k_1 represents a reaction at a low temperature and k_3 a reaction at higher temperatures. [56]

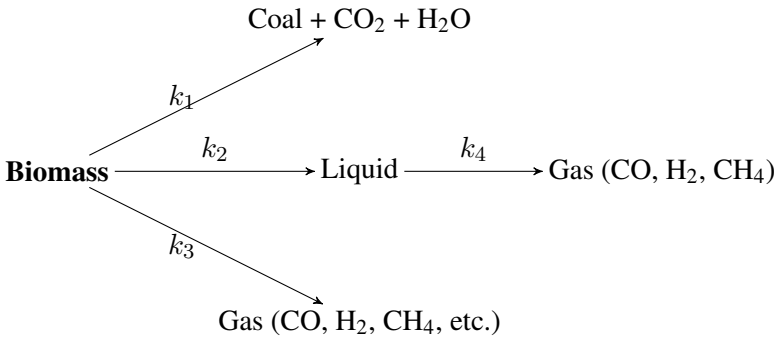


Figure 2.13: Typical thermochemical lignocellulosic biomass decomposition according to Kaltschmitt et al. [56]

- **Gasification**
  Solid, liquid and gaseous pyrolysis products are forced to further reactions through further increase of the heat level and addition of oxidant. Oxygen in the oxidant (steam, air, pure oxygen, carbon dioxide) is required for the gasification of solid pyrolysis products (pyrolysis coke). The penetration of biomass particles by the oxidant is hindered through outgasing pyroylsis gases, therefore oxidants do not get access to pyrolysis coke during the pyrolysis phase. Consequently, gasification cannot start before the pyrolysis has finished, at least for small particles. For bigger particles gasification already starts parallel on the surface while pyrolysis is still happening in the particles core. Above about 600°C, gasification already starts during the pyrolysis process because oxygen is embedded within biomass (H_2O, CO_2,...). In dependency of the temperature, pressure and the amount of oxidant, solid-gas gasification reactions are occurring. They are described by Equation 2.9 to 2.13. [56]

  Full oxidation:

$$C + O_2 \rightarrow CO_2 \qquad \Delta H = -393,5 \frac{kJ}{mol} \tag{2.9}$$

  Partial oxidation:

$$C + \frac{1}{2} O_2 \rightleftarrows CO \qquad \Delta H = -110,5 \frac{kJ}{mol} \tag{2.10}$$

  Heterogeneous water-gas reaction:

$$C + H_2O \leftrightarrows CO + H_2 \qquad \Delta H = +118,5 \frac{kJ}{mol} \tag{2.11}$$

  Boudouard reaction:

$$C + CO_2 \leftrightarrows 2\ CO \qquad \Delta H = +159,9 \frac{kJ}{mol} \tag{2.12}$$

  Hydrogasification:

$$C + 2\ H_2 \leftrightarrows CH_4 \qquad \Delta H = -87,5 \frac{kJ}{mol} \tag{2.13}$$

The gaseous products of solid-gas reactions and pyrolysis gas are further treated by either heterogeneous catalytic or homogeneous gas-gas reactions:

Water gas shift reaction (WGS):

$$CO + H_2O \leftrightarrows CO_2 + H_2 \qquad \Delta H = -40,9\frac{kJ}{mol} \tag{2.14}$$

Methanisation reaction:

$$CO + 3\ H_2 \leftrightarrows CH_4 + H_2O \qquad \Delta H = -203,0\frac{kJ}{mol} \tag{2.15}$$

Reforming of hydrocarbons:

$$C_mH_n + mH_2O \leftrightarrows mCO + (m + \frac{n}{2})H_2 \qquad \Delta H > 0 \tag{2.16}$$

The product gas composition of the gasification process cannot be derived just by the mentioned chemical reactions. In reality, chemical equilibrium is not reached. It is also necessary to consider gas/solid transportation processes, reactor design and the presence of possible catalysts.
In order to get as much as possible of the desired product gas, operation conditions have to be adjusted. Table 2.3 presents a short summary of parameter influence for selected equations on equilibrium floating conditions (EFC).

Equation	Temperature	Pressure	EFC shift in direction of:
2.11	↑	↓	CO and H_2
2.14	↑	-	CO and H_2O
2.15	↓	↑	CH_4

↑ increasing parameter, ↓ decreasing parameter, - parameter has no influence

Table 2.3: Influence of changing parameters according to EFC [56]

2.3.2 Gasification reactors

Gasifiers are characterized according to the contact between fuel- and oxidant- respectively the product-gas. The following reactors are well established for gasification:

- Fixed bed reactors
- Fluidized bed reactors
- Entrained flow reactors

In terms of biomass gasification only the two last options are of interest and discussed.

Fluidized bed reactors

In order to create a fluidized bed environment with an inert or catalytically active material (bed material) the carrier gas (oxidant) has to enter the reactor within a certain velocity range. Depending on this velocity, the bed material is either in a floating- (stationary fluidized bed) or in a circulating-state (circulating fluidized bed). Due to a good mixing of bed material and solid fuel, a homogeneous temperature- and reaction-zone is achieved. This leads to an intensive heat transfer between the bed material and the solid biomass. Hence, the residence time in the reactor can be restricted to minutes or even seconds. Comparing to other gasification technologies, the main advantages of this type of reactor are the ability to handle a wide range of feedstocks with a high moisture content, its constant temperature level (700-900°C) and its relatively easy adjustability. [25][56]

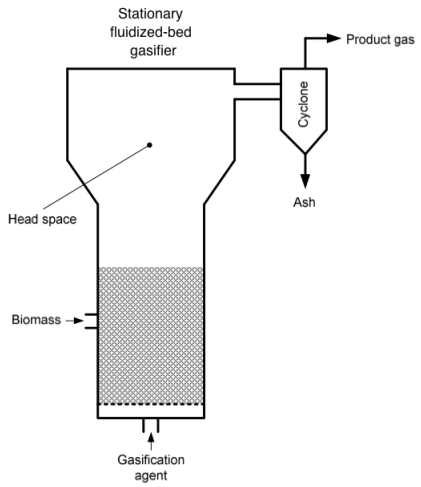


Figure 2.14: Stationary fluidized bed gasifier [30]

Stationary (bubbling) fluidized bed reactor, from the bottom of the reactor the gasification agent enters the gasifier fast enough to keep the bed material (e.g. olivine) in a floating position but does not make it leave the fluidized bed (Fig.2.14). Next to the fluidized bed zone, pyrolysis gases and ash are entering the head space (freeboard). The freeboard temperature is 10-50°C lower than within the fluidized bed zone. Low ash melting behavior can be a problem of the fluidized bed technology and in general

fluidized bed biomass gasification is characterized by a high amount of fly ash. Remaining solid particles in the product gas have to be removed by a cyclone, fabric filter or other separation options. Compared to a fixed bed reactor, feedstock can be gasified without slag building because of the lower and constant operating temperature. Another advantage is its flexibility with respect to changes in feedstock composition (moisture, ash). [56]

Circulating fluidized bed reactor, the principle for the circulating fluidized bed is the same as the one of a bubbling fluidized bed reactor except for the fact that there is no longer a bed zone and a head space. In comparison to the bubbling fluidized bed technology, the circulating fluidized bed gasification requires a higher gasification agent flow velocity (>10 m/s). The bed material is homogeneously distributed over the whole reaction chamber. It is also a bit finer compared to the stationary fluidized bed reactor. In a continuous process, the bed material leaves the gasifier with the product gas and enters again after gas-solid separation by a cyclone (Fig.2.15). [56]

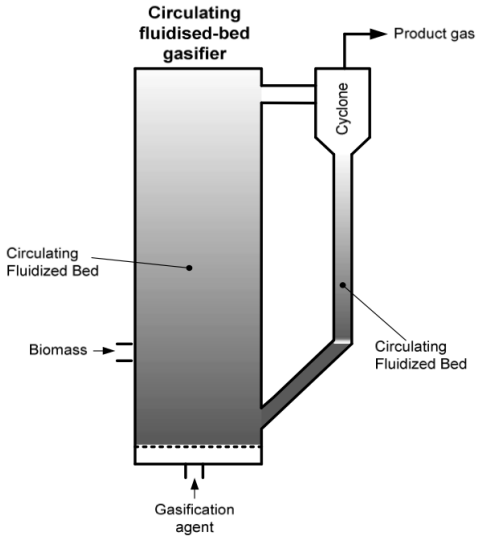


Figure 2.15: Circulating fluidized bed gasifier [30]

Dual fluidized bed (DFB) reactor, in order to achieve a desired product gas quality, a combination of more than one fluidized bed reactors is utilized. One reactor is used for the gasification of feedstock and the other one provides the required energy for the endothermic gasification reactions via combustion of a small fraction of the feedstock and/or additional fuel and/or recycled flue gas. For the concept of Section 3.1 an OxyFuel process is installed for the combustion chamber due to the availability of oxygen. The heat transport from the combustion reactor into the gasification reactor can either be done by a circulating heat carrier or by a high temperature heat carrier in a heat exchanger. [56]

- Circulating heat carrier
  Heat is transferred from the combustion to the gasification chamber through bed material (e.g. Fig.2.18). The advantage of this system is the lower investment costs for the transportation material in comparison to a high temperature heat carrier. The technology has reached state of the art status.[56] Several gasifiers were operated with this technology for years e.g. Güssing, Austria.[50]

- High temperature heat carrier
  In the combustion chamber a high temperature heat carrier (e.g. sodium, potassium) is vaporized. The gaseous heat carrier is transported within closed heat-pipes to a cooling zone where the heat-pipes are surrounded by bed material of the gasification chamber. The gaseous heat carrier releases the energy to the bed material and condenses. This principle is interesting for small capacity applications. Further research considering its long term reliability is necessary. [56]

Two special operation methods of the DFB gasification with a circulating heat carrier -developed at TU Wien- are presented below. Both operation methods are using the same gasifier design but different operation conditions and bed material.

- **DFB-process**
  This process has been developed at TU Wien and successfully utilized for years at the demonstration power plants in Güssing and Oberwart, Austria. The dual fluidized bed (DFB) gasifier with olivine as bed material is the core module of this power plants. [50] A scheme of the reactor is shown in Figure 2.16. Steam acts as oxidant for the gasification of biomass in a stationary fluidized bed reactor. Consequently, almost no nitrogen is in the medium calorific (12-14 MJ/Nm^3 dry gas) product gas and the heating value is higher in comparison to the application of air as gasification agent.
  The required heat for gasification is provided by the circulating, heat carrying bed material. This is an allothermic gasification process, in comparison to an autothermic process where all the required heat would be produced by partial combustion of biomass (pyrolysis coke) within the gasification chamber. The bed material leaves the gasification reactor at the bottom through a chute to the combustion reactor. Both gasification-combustion connections (chute and loop seal) are fluidized by steam to avoid gas leakage and provide a high throughput. If the gasification temperature drops down, the residual amount of ungasified biomass increases and provides more fuel for combustion in the circulating fluidized bed. This leads to an increase of heat transfer to the bed material and therefore an increase of heat added to the gasification zone. The system is auto-stabilized, however, the temperature cannot just be reached by char from the gasification reactor on its own. Additional fuel for the combustion reactor is necessary.

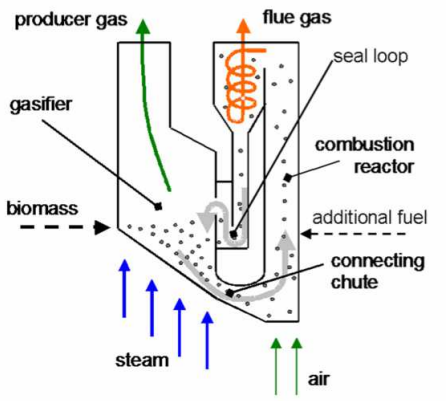


Figure 2.16: DFB scheme [75]

Table 2.4 shows a typical product gas composition for DFB-gasification of biomass. All quantities except the water concentration are related to the dry gas fraction. For further detail see [50].

Species		Unit
H_2O	30-40	vol.%
CH_4	10-11	vol.% (dry)
C_2H_4	2-2.5	vol.% (dry)
C_3 fraction	0.5-0.7	vol.% (dry)
CO	24-26	vol.% (dry)
CO_2	20-22	vol.% (dry)
H_2	38-40	vol.% (dry)
N_2	1.2-2.0	vol.% (dry)
H_2S	130-170	vppm (dry)
NH_3	1100-1700	vppm (dry)
Tar	2-5	g/Nm^3 (dry)
Particulates	20-30	g/Nm^3 (dry)
LHV	12.9-13.6	MJ/Nm^3 (dry)

Table 2.4: Typical composition of product gas from the DFB gasification process before gas cleaning [77]

Figure 2.17 shows the Oberwart CHP plant. The gasification unit in Oberwart has a thermal power is 8.6 MW. [16] Besides the DFB unit and the related product and flue gas cleaning, heat recovery through an ORC unit and two gas engines for the conversion of product gas into electricity are installed.

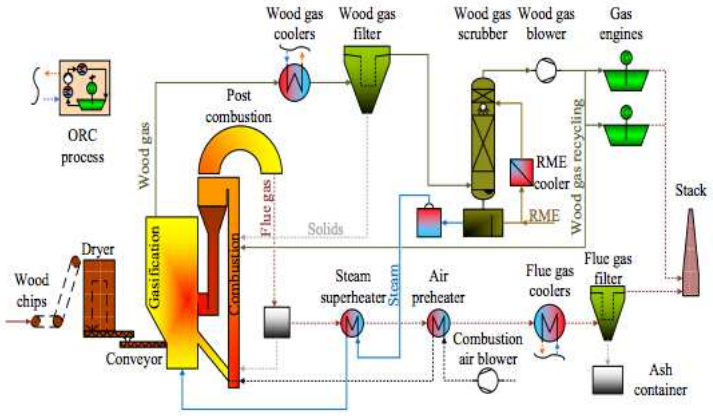


Figure 2.17: Process flow scheme of the CHP plant in Oberwart, Austria [16]

In Table 2.5 significant values for biomass gasification are summarized. The table shows slightly higher temperatures than discussed in theory.

Parameter	
Gasification temperature	846 - 848°C
Combustion temperature	930 - 945°C
Steam fluidization	450 - 690 kg/h
Steam-to-fuel ratio (waf)	0.5 - 0.6 kg/kg
Specific gas production (dry gas, dry fuel)	0.9 Nm^3/kg
PG temperature before filter	130 - 140°C
PG temperature after scrubber	36 - 37°C
FG temperature before filter	130 - 144°C

Table 2.5: Typical operation conditions for a DFB gasifier [110]

- **SER-process**
  SER is the shortcut of sorption enhanced reforming (SER). It is a variant of the DFB process for a maximum H_2 content in the product gas. The bed material olivine (DFB) is replaced by CaO (SER - Fig 2.18). The technology requires a low temperature level in the gasification reactor, so the bed material circulation has to be reduced. This is maintained through an increase of the bed material's particle size and a decreased of the primary fluidization in the combustion reactor. [75]
  At atmospheric pressure and the right temperature level (Fig. 2.19), additionally to the procedure of the DFB-process, in-situ sorption of CO_2 takes place while gasification. Carbon dioxide from the gasification process is absorbed respectively adsorbed exothermic at 600-700°C by the bed material in order to get a lower CO_2 content in the product gas. Also the heat demand is lowered for the gasification chamber because of the released energy from exothermic CO_2 sorption. The following reactions play an important role for SER.

$$CaO + CO_2 \leftrightarrows CaCO_3 \qquad \Delta H = -181,4\frac{kJ}{mol} \tag{2.17}$$

$$CO + H_2O \leftrightarrows CO_2 + H_2 \qquad \Delta H = -40,9\frac{kJ}{mol} \tag{2.14}$$

  Due to CO_2 sorption a CO-shift occurs during gasification, leading to a higher H_2 content in the product gas. The separation of gasification- and combustion-part enables an individual, optimal design for both of them. Therefore, a higher product gas quality is achieved and post gasification cleaning costs are reduced. CaO is a good catalyst/bed material for the prevailing gasification conditions of 600-700°C at atmospheric pressure. The low temperature level might guide to the assumption of a low carbon conversion rate. However, the actual efficiency of an SER-gasifer is higher compared to conventional gasifiers (see Table 2.6). At a temperature of about 800°C in the combustion reactor $CaCO_3$ is revitalized through desorption of CO_2.

Component	Conventional process[a]	SER-process[b]
H_2O, vol%	30…45	51...65
CH_4, vol%db	10…11	10…14
C_2H_4, vol%db	2…2.5	1…2
C_3-Fract., vol%db	0.5…0.7	0.5…0.8
CO, vol%db	24…26	4…8
CO_2, vol%db	20…22	6…13
H_2, vol%db	38…40	65…75
H_2S, v-ppm db	130…170	11…37
NH_3, v-ppm db	1100…1700	840…965
Tar g/m^3_n db	2…5	0.5…3.5
LHV MJ/m^3_n db	12.9…13.6	13.1…16.5

[a] $8MW_{th}$ demonstration plant in Güssing, Austria according to Pröll et. al., 2005
[b] $100kW_{th}$ process development unit at the Vienna University of Technology, Austria

Table 2.6: Comparison of product gas composition for wooden feedstock (Conventional:Wood chips, SER: Wood pellets) [75]

In order to properly interpret the comparison of Table 2.6, it has to be mentioned that the hydrogen content mostly is a function of the temperature and the biomass to steam ratio. For the comparison of Table 2.6 all these parameters are kept equal. Hydrocarbon contents are relatively independent from the process conditions for both, the conventional and the SER process. H_2O, H_2, CO_2 and CO are closely linked to the CO_2 sorption and shift equilibrium. A detailed description of the 100 kW SER gasifier and its process conditions can be found in the paper of Pfeifer et.al.[75].

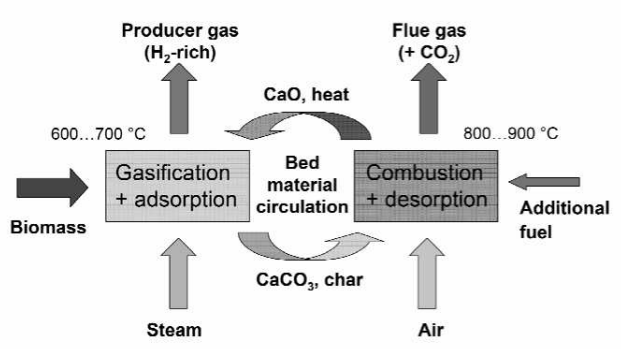


Figure 2.18: In-situ adsorption in the SER concept [75]

For the considered temperature range of 800-900°C, within the combustion chamber, $CaCO_3$ is split into CaO and CO_2 exothermic (Fig. 2.18). [75]

If necessary, additional fuel (e.g. recycled producer gas, saw dust, etc.) can be added to regenerate and heat up the bed material in the combustion chamber. Before the flue gas leaves the gasifier, the heated bed material is separated in a cyclone and returns via a loop seal into the gasification reactor.
The relevant partial pressure, for proper CO_2 removal through the flue gas, for gasification is of about 0.25 MPa and for combustion of about 0.1 MPa (Figure 2.19).

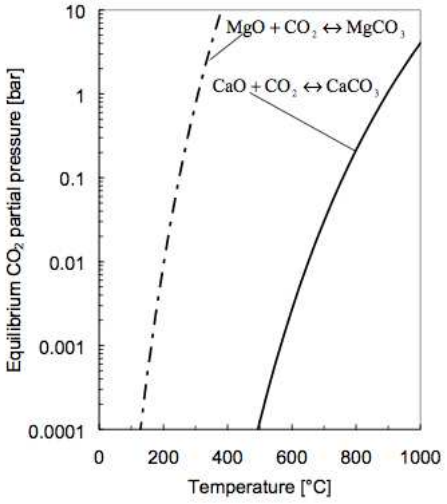


Figure 2.19: Equilibrium partial pressure of CO_2 for CaO and MgO, respectively [75]

Entrained flow reactors

For this type of gasifier, a low tar content in the product gas and a high char conversion rate are characteristic. This is related to the high gasification temperature of 1200-1500°C of the finely pulverized fuel. The fuel pretreatment is the main reason why this technology is not useful for biomass gasification but for coal. Getting biomass finely distributed is too expensive for commercial scale and it is also hard to achieve the characteristic high temperature because of biomass' low energy density. [25]

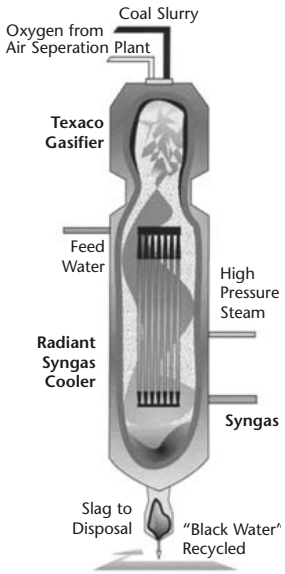


Figure 2.20: Entrained flow gasifier [17]

According to the Research Center Karlsruhe -nowadays known as Karlsruhe Institute of Technology (KIT)-, a higher energy density should be obtained by creating a slurry out of biomass, tar-oil and coke. A typical installation size of several hundreds of MW or even GW is reasonable as the pressure operation allows a very small reduced volume for the gasifier. [17]
Figure 2.20 shows an entrained flow gasifier for char slurry by the company Texaco. The technology was bought in 2004 by GE Energy. There are two different ways of running

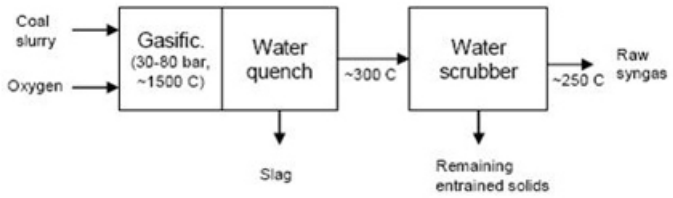


Figure 2.21: Entrained flow gasifier using direct quench mode [71]

this gasifier. Either the syngas can be cooled by a radiant and/or convective heat exchanger and/or via a direct quench system. For the direct quench system (Fig.2.21), water or cooled recycle gas is added to the hot syngas.
For the setup represented by the scheme of Figure 2.21, the product gas is immersed in water because of a quench ring. The direct contact with water decreases the efficiency but is preferred for industrial use as it is less expensive and better considering its operation reliability. The gasifier setup for the scheme of Figure 2.22 includes a soot-tolerant radiant

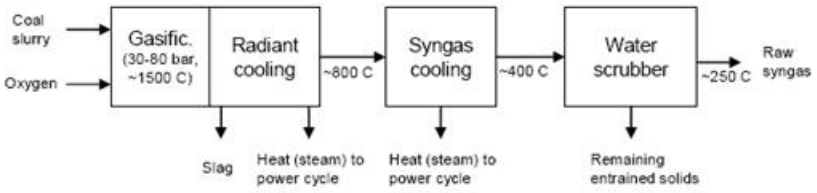


Figure 2.22: Entrained flow gasifier using radiant heat exchanger [71]

syngas cooler that creates high pressure steam while it is cooling the product gas. This increases the gasifier's efficiency. At its bottom a water pool is located where slag is quenched and removed via a lock hopper. This constellation leads to an increase of CO within the product gas stream. After leaving the gasifier, residual particles and char are removed by a water scrubber.[71]

2.4 Gas cleaning

Gas cleaning is necessary upstream a methanation reactor as e.g. Ni-based catalysts are applied and have to be protected from harmful components. Those components and their upper level are listed in Table 2.12. General information on catalyst deactivation can be found in Subsection 2.5.1.
The gas cleaning design for the Power to Gas scenario with carbon supply through DFB gasification (Section 3.1) is visualized in the process flow scheme of the overall process,

Figure 3.2. The second Power to Gas scenario just downstream SNG purification by a membrane is considered (Section 3.2). This is because of the assumption of pure CO_2 and pure H_2 as methanation feed gas.
In the following only these devices are introduced, which are actually used for the simulation.

2.4.1 Particle removal - Fabric Filter

Particle removal can be done by gravity and centrifugal separators, filter with adhesive forces and filter effect, separation through electric fields or wet dust extraction.
In the context of this disseratation's biomass gasification -the second option- a fabric filter has to clear the product and flue gas from solid particles as bed material, ash or char. Its operation condition are listed in Table 2.7. Due to an operation temperature of 130-200°C, middle- to high-boiling tars should be removed as well.[20] In this case so called precoated filter are applied.
Further information about fabric filter can be found for example in the book Air Pollution Control Engineering [107].

Parameter	Range
Temperature	<180°C
Particle size for selective removal	0.5-100 μm

Table 2.7: Influencial parameters for fabric filters [56]

2.4.2 Tar removal - RME scrubber

In general tar removal can be done by a washer, electrostatic precipitator or through thermal or catalytic conversion into stable components, like in the gasification reactor. In this context the separation through RME-scrubbing is highlighted.
The typical gas composition of product gas from a DFB-gasifier is described in Subsection 2.3.2, Table 2.4. Besides the desired components (H_2, CO, CH_4) also unwanted fractions as tar, sulphur or nitrogen are present in the product gas. Tar removal through sorption by an organic solvent is highly effective. In case of the CHP power plant Güssing the applied solvent is rapeseed methyl ester (RME). In the work of Pröll et.al. [77] the results for the gasifier's product gas cleaning are presented. The results show a positive side effect, the condensation of water in the scrubber unit. Through the condensing water, also trace components (mainly NH_3 and H_2S) are partly removed. In case of ammonia (NH_3) a removal of up to 50% can be expected, for the gas composition of Table 2.4. [77]

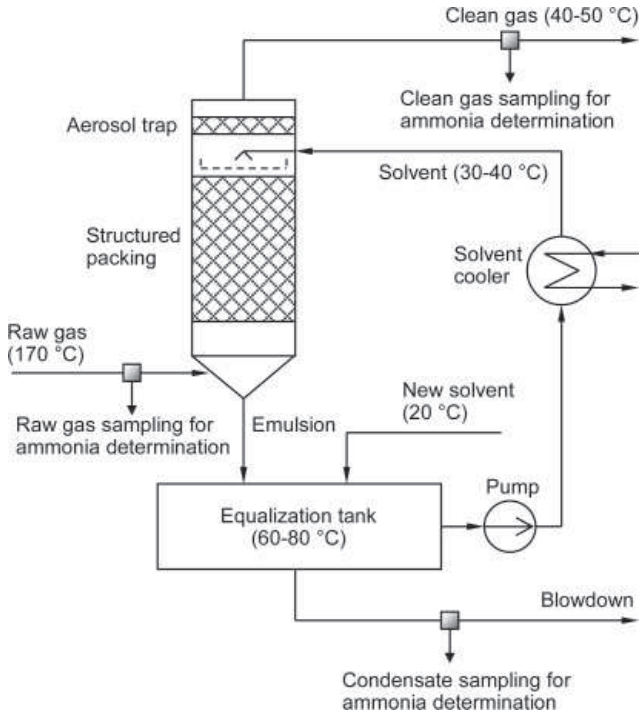


Figure 2.23: Scrubber setup at the Güssing CHP-plant, Austria [77]

Figure 2.23 shows the scrubber design for the CHP-plant Güssing. The removal of nitrogen- and sulphur-components is relevant as they are causing damage of downstream applications as gas engines or methanation catalysts. [77]
General operation parameters for scrubber application are shown in Table 2.8.

Parameter	Range
Condensation temperature in purified gas	down to -17°C
Tar content purified gas	20-40 mg/m_3
Pressure drop	30-2008 mbar
Temperature	<100°C

Table 2.8: Influencial parameters for scrubber application [56]

2.4.3 Sulphur removal - Activated char coal filter

Downstream the RME-scrubber, further product gas treatment is necessary. For sulphur removal the application of activated charcoal is a state of the art technology. It is operated at atmospheric pressure. According to research of [41], the activated charcoal unit is sufficient to clean the gasifier product gas from sulphuric components below the detection limit, see Figure 2.24. As described in Subsection 2.5.1, the sulphur limitation for methanation is equal to the one of FT-synthesis.[81] Therefore, the results for sulphur removal in a 1000 hours longterm test in May 2007 at TU Wien can be applied for methanation as well, see

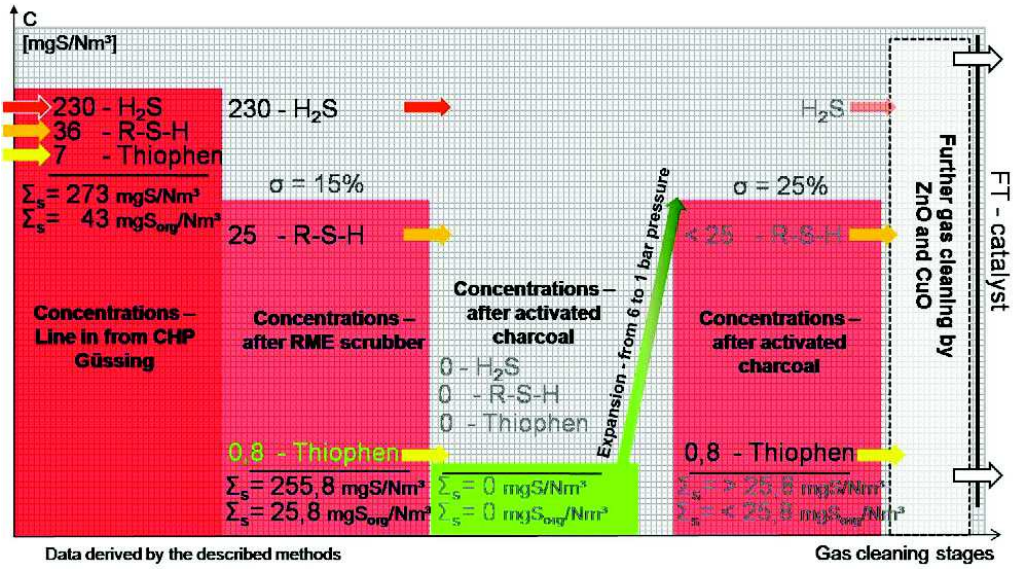


Figure 2.24: Schematic balance of sulphur contaminates after wet gas cleaning and the activated charcoal [41]

[41]. Due to a considered expansion, sulphur is released in this test run after the activated char coal filter. So, additionally to RME-scrubber and activated charcoal filter a zinc-oxide filter was installed. In this dissertation's simulation no expansion is expected and therefore no zinc-oxide filter is included.
It was proven that the chosen setup does not cause any catalyst damage for the FT-catalyst.[41]
The overall reaction for H_2S removal can be described as [33]:

$$H_2S + 0.5 \cdot O_2 = S + H_2O. \tag{2.18}$$

Sulphur remains adsorbed on the activated charcoal until the surface is fully covered, afterwards the adsorbends is revitalized. Besides H_2S also mercaptanes, other sulpuhr components and ammonia are adsorbed by the impregnated adsorbens. [33][28]
Table 2.9 shows some specific values for impregnated activated charcoal.

Parameter	Range
Pressure drop	0.5-30 mbar
Temperature	10-70°C
H_2S Breakthrough capacity	12 g H_2S/cm^3 activ. charcoal

Table 2.9: Parameters for activated charcoal application [33]

2.4.4 Hydrogen recovery - Membrane

For SNG purification, gas permeation membranes are applied. Besides this technology reverse osmosis, nanofiltration, ultra- and microfiltration, electrodialysis, pervaporation are done with a membrane but also membrane contactors and reactors are available. [65] The great interest in membrane gas separation is caused by its beneficial investment costs and almost no operation costs.
The gas transport velocity through the membrane is dependent on the process parameter on permeate-, retentate- and feed-side, as well as interaction between the transported gas and membrane material. In general the membrane is either a porous- or polymer-membrane. In any case the transmembrane flow J of the component i is defined after Equation 2.19. The membrane area is described by the variable A, P_i is the permeance of the component i and the pressure term describes the partial pressure difference of the component i between the feed and permeate stream. [16]

$$J_i = A_m \cdot P_i \cdot (p_{i,F} - p_{i,P}) \tag{2.19}$$

Porous membrane units are acting like a sieve, simply put. Particles smaller than a certain size are permeating through the membrane and those too large are leaving through the retentate exit.

Solution diffusion membrane units are more complex. The permeating species enters the membrane through sorption. First it is dissolved from other species, if it is bonded to them. Afterwards it is diffusing through the membrane. [66]

Membrane type	Advantages	Disadvantages
Organic	• Industrial familiarity • Ease of manufacture and installation • Controllable pore size distribution, thickness and geometry	• Can't withstand all solvents and strong acids or bases • Impractical for high temperature separations. Must generally be less than 200°C
Metal	• Well established technology	• Expensive • Poor mechanical stability
Zeolite	• Separations possible at higher temperatures • Control of pore sizes	• Pore sizes general too large for selective CO/H_2 separation
Ceramic (non-silica based)	• Separation in more demanding environments possible	• Low fracture toughness • Complicated processing steps • Processing techniques not well suited for microporous materials
Ceramic (silica based)	• Processing suitable for microporous materials • Controllable pore size distribution, thickness and geometry • Higher temperature resistance	• Low H_2 permeability • Sensitive to H_2O vapour (irreversibly densifies structure)

Table 2.10: Advantages and disadvantages of various types of membranes

The advantages and disadvantages of the various membrane materials are listed Table 2.10. For the product gas upgrade downstream the methanation reactor a polymer membrane has

been chosen (Section 3.2). Further details on polymer membranes can be found in [65].

Typical membrane specifications for SNG purification by a hollow fibre membrane can be found in Table 2.11. The applied temperature is individual and related to the used membrane material. For polycarbonate e.g. it is 140°C, this and temperatures of further materials for SNG purification can be found in [46].

Parameter	Range
Feed pressure	about 30 bar
Permeate pressure	1 bar
Pressure drop flow related, feed	<2 bar
Pressure drop flow related, permeate	1-5 bar

Table 2.11: Operation parameters of a hollow fibre membrane for SNG purification [65]

2.5 Methanation

In order to use the enormous capacities of the natural gas grid, excess electricity and carbon-monoxide/dioxide are converted into synthetic natural gas SNG. The conversion is described through Equation 2.20 to 2.22. The reaction enthalpies ΔH are defined for standard conditions.

$$CO + 3H_2 \rightleftharpoons CH_4 + H_2O \qquad \Delta H = -206kJ/mol \tag{2.20}$$

$$CO_2 + 4H_2 \rightleftharpoons CH_4 + 2H_2O \qquad \Delta H = -165kJ/mol \tag{2.21}$$

$$CO + H_2O \rightleftharpoons CO_2 + H_2 \qquad \Delta H = -41kJ/mol \tag{2.22}$$

According to [97], CO-methanation (Eq.2.20) occurs first and, if excess hydrogen is left, CO_2-methanation (Eq.2.21) starts. CO_2-methanation can be seen as combination of CO-methanation (Eq.2.20) and the water gas shift reaction (Eq.2.22).
Methanation is a volume lowering, exothermic process. Therefore, the chemical conversion increases via lowering the temperature and increasing the pressure. The methanation's lower temperature level is located at roughly 200°C. Below this temperature nickel carbonyl is formed due to the applied catalyst, see Subsection 2.5.1. Above a temperature of 550°C, sintering of the catalyst begins and methane will be reformed into its reactants (natural gas steam reforming reaction). [10]

2.5.1 Catalyst types

The methanation process is pushed heterogeneously by a solid catalyst. Basically, all transition elements of the 8th sub-group can catalyse methanation reactions. But also the WGS-reaction is catalysed by them. For industrial application in most cases nickel or ruthenium based catalysts are applied whereas nickel is favoured due to its advantages considering activity, selectivity and price. Besides oxygen mainly sulphur has harmful effects on these catalysts. [11]
The main reasons for catalyst deactivation are, according to [67]:

- Chemisorption of foreign objects on catalytic active centres (catalytic toxication)
- Covering of catalytic active surface or blockage of pore volume by solid particles (e.g. char)
- Conversion of metallic catalysts into gaseous phase (e.g. formation nickeltetracarbonyl $Ni(CO)_4$
- Thermal sintering

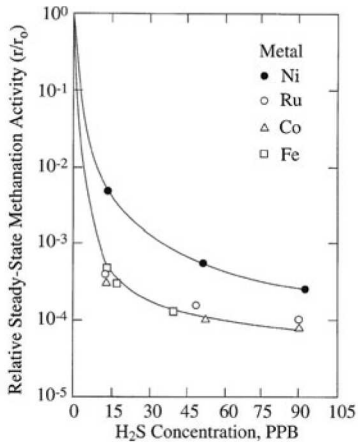


Figure 2.25: Relative methanation activities at equilibrium state for nickel(Ni), cobalt (Co), iron (Fe) und ruthenium (Ru) in dependency of the concentration of H_2S in gaseous phase. Reaction conditions: 100 kPa; 400°C; 1% CO, 99% H_2 for cobalt, iron and ruthenium; 4% CO, 96% H_2 for Nickel [14]

Figure 2.25 shows that a H_2S content of 15-100 ppb is sufficient to lower the activity (r/r_0) of the initially unpolluted catalyst (r_0) by 3-4 orders of magnitude. According to [67] the toxicity of sulphur compounds is ranked as $H_2S > SO_2 > SO_4^{2-}$.
If nickel turns into gaseous phase, the catalyst can be blown out of the fluidized bed reactor due to its lower weight. Further, nickeltetracarbonyl is highly toxic for the human organism. Covering but also sintering are causing major loss of catalytically active surface.
The restricting impurity levels are defined by Table 2.12. In the work of [81] it is assumed that the acceptable impurity levels for methanation are equal to those of Fischer-Tropsch synthesis, as similar catalysts are used. For FT-synthesis mainly Co-based catalysts are

chosen which are quite sensitive considering sulphur as well [111]. According to [83] the upper level for H_2S in the German natural gas grid is 6 mg/m^3, this roughly equals 4 ppm or 0.0004 mol.% H_2S.
Further information considering catalyst requirements and characteristics for methanation can be found in [83].

Impurity	Removal level
Sum of sulphur compounds (H_2S+COS+CS_2)	$<$ 1 ppmV
Sum of nitrogen compounds (NH_3+HCN)	$<$ 1 ppmV
HCl + HBr + HF	$<$ 10 ppbV
Alkaline metals	$<$ 10 ppbV
Solids (soot, dust, ash)	essentially completely
Organic compounds (hydrocarbons, tars)	below dewpoint

Table 2.12: Maximum impurity levels for Ni-based catalysts for FT or SNG synthesis [19]

2.5.2 Reactor types

In the following it is differed between fixed bed, fluidized bed and alternative reactor concepts. For the first two traditional concepts related projects are listed either operated with synthesis gas from coal or biomass.

Methanation subject to fixed bed reactors

Fixed bed reactors perform a heterogeneous methanation. The reactants are in gaseous phase while the catalyst is in solid phase. The catalyst size is in millimeter range. It creates a fixed bed within the reactor that the gas is floating through from top to bottom. The main advantage is the low mechanical stress for the catalyst, in comparison to a fluidized bed. The major disadvantage is caused by the relatively large particle size which restricts the mass transport. [97]
Fixed bed reactors are considered as adiabatic. Due to the restricted heat removal ability within the fixed bed, local hot spots can be built which cause catalyst damage and therefore lower conversion rates [11].
To maintain a high conversion rate at isothermal conditions, great effort is required. The main purpose is to install several conversion reactors with low conversion rates so that less heat is produced in each of them. They can either be connected serial or parallel, whatever is best for the specific case. Another approach is to cool the gas stream between several reactors or to use inert recirculation gas. In case of intermediate cooling, the product gas of each reactor is cooled with a heat exchanger. The obtained excess heat is often used for steam production. When using recirculation gas its temperature is first decreased in a heat exchanger as well. Via the recirculating gas, the reactant gas stream gets diluted. Due to the option of dilution, more heat can be removed from the reactor and therefore almost isothermal conditions can be created [97]. A comprehensive overview of all the advantages and disadvantages as well as their current state of development is presented in the work of

Rönsch and Ortwein [97] for all the processes related to coal and biomass.

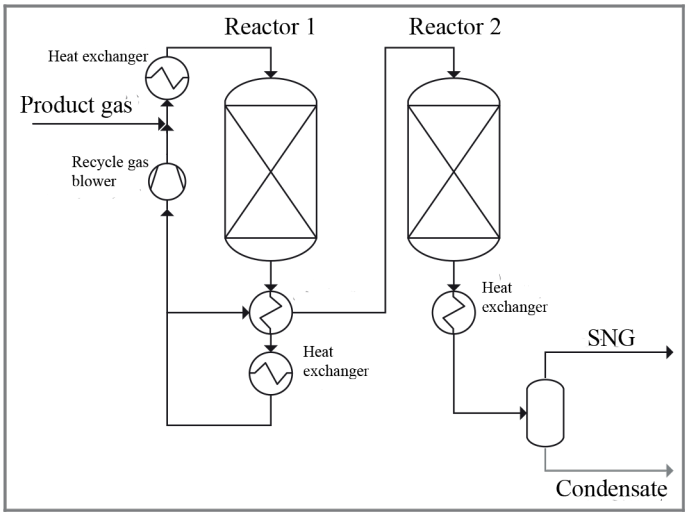


Figure 2.26: Schematic installation of Lurgi methanation [97]

For the Lurgi methanation process (Figure 2.26) both options, intermediate cooling and dilution, are considered.

	Developer	Feed-Typ	Process-Steps	Pressure [bar]	Temperature [°C]	Year
TREMP	Haldor/Topsøe	Coal	3	30	300-700	1980
Hicom	British Gas Corp./Lurgi	Coal	4	25-70	230-640	1981
RMP	Ralph M. Parson Co.	Coal	4-6	1-70	315-780	1974
Hygas	IGT	Coal	2	70	280-480	1955
ECN		Biomass	2	18	450	1974
ZSW		Biomass	1	86	k.A.	1965

Table 2.13: Overview on fixed bed methanation processes subject to fixed bed reactors [11][59]

Table 2.13 shows different commercially applied fixed bed methanation methods. The methods can be further separated by their application for coal or biomass based synthesis gas methanation.

Coal based methanation means that methanation's feed gas is created via coal gasification. This concept was popular in the 1970s when the natural gas consumption was increasing due to the oil crises. Most of the concepts of Table 2.13 were developed at this time. The interest in natural gas substitutes decreased when the energy price decreased again. A schematic illustration of the methanation processes RMP, TREMP and HICOM can be found in the work of Rönsch and Ortwein [97].

Biomass based methanation has been considered due to the growing interest in sustainability. When using biomass, the methanation units are smaller in size but also different considering their technology and economic situation. The *Energy Research Center of the Netherlands (ECN)* has done experimental work at lab scale [8]. The reactor concept of the ECN is called ESME and represents a serial connection of fixed bed reactors quite similar to conventional reactor systems [79].

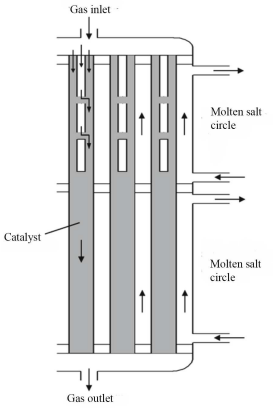


Figure 2.27: Schematic installation of the tube boundle reactor using two separated cooling circles and staged reactants injection [83]

Also the *Zentrum für Sonnenenergie - und Wasserstoff-Forschung (ZSW)* has done research in the field of fixed bed methanation. They are using tube bundle and plate reactors for cooling. For the tube bundle reactors, the tubes are filled with the catalyst and these are surrounded by the cooling agent, which is molten salt. At the reactant gas inlet, more reactants are available for synthesis and so the temperature is at its peak at the inlet, due to the exothermic methanation reactions. In order to avoid a hot spot at the upper gas inlet, more gas inlets are installed downstream (Figure 2.27). [59][83]

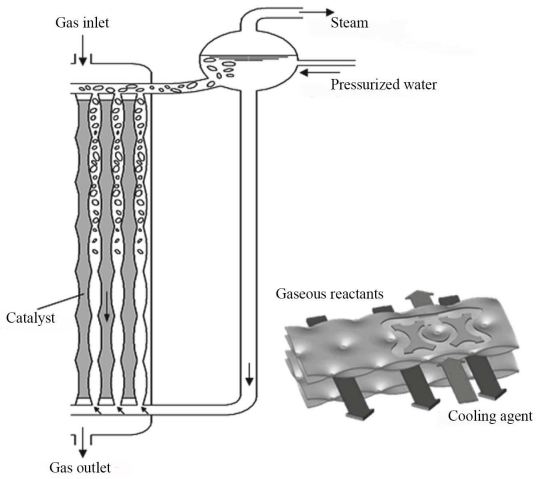


Figure 2.28: Schematic installation of water/steam cooled plate reactor [83]

For plate reactors (Figure 2.28), the cooling agent is floating within the cavity space and is surrounded by the catalyst. As cooling agent pressurized water has been chosen. Within the plates, water is partially vaporized and further separated from steam, as presented in Figure 2.28. The methanation product gas composition is controlled by the water pressure and therefore the evaporation temperature. The water pressure is set in a way that at the gas outlet an ideal temperature is obtained to achieve the desired product gas composition.

A main advantage of both concepts (*ESN/ZSW*) is the excellent upscaling predictability. When knowing the properties of one tube or plate, upscaling can be done by duplicating the installation as many times as required. There is no need for a connection of several reactor units, the whole methanation can be done within one reactor which is especially desirable considering the costs.[59][83] The overall research projects of *ECN* and *ZSW* for SNG production are described in Section 1.2.

Methanation subject to fluidized bed reactors

In comparison to a fixed bed reactor, catalyst particles are smaller in size. This has positive aspects on the mass transfer due to a higher specific catalyst surface. The removal of reaction heat can be seen as the main advantage in comparison to fixed bed systems. Fluidized bed reactor methanation can be seen as an isothermal process. Its major disadvantage is the high level of mechanical catalyst stress. This could end up in destruction of the catalyst.
For fluidization, a certain gas flow has to be injected into the reactor. The minimum and maximum fluidization gas flow rates are limited by the minimum fluidization- and the particle entrainment-point [11]. Considering fluidization- and the particle entrainment-point see [51].

	Developer	Feed-Typ	Process-Steps	Pressure [bar]	Temperature [°C]	Year
Lurgi/ Sasol	Lurgi	Coal	2	18	450	1974
Bi-Gas	Bitumious Coal Research Inc.	Coal	1	86	k.A.	1965
Comflux	Thyssengas/ EBI (PSI)	Coal	1	20-60	400-500	1980 (2008)
COALA/COSYMA	PSI/TU Wien	Biomass	1	2.9	200-300	2007-2009

Table 2.14: Overview of methanation processes employing to fluidized bed reactors [11][59][81][58]

Coal based methanation, in this context the Bureau of Mines project, Bi-Gas project and the Comflux process are worth mentioning. In the order as listed, the first program started the early 50s, through the 60s and the last one from the 70s till the 80s. Detailed information can be found in [58].

Biomass based methanation was improved significantly by the Güssing bioSNG project [82]. The project was initiated by the PSI-Paul Scherrer Institut (CH), TU Wien

(A), CTU-Conzepte Technik Umwelt AG (CH) and Repotec Renewable Power Technology Umwelttechnik GmbH (A). BioSNG was produced from a product gas share of the 8 MW biomass gasification demonstration plant. The methanation reactor design was done by PSI (Figure 2.29). [89][90]
In the gasifier product gas unsaturated hydrocarbons are present, in contrast to other sources. Mainly C_2-hydrocarbons are present, they can cause carbon deposits and nickel detachment on the catalyst's surface and therefore its deactivation. The level of dis-activation is dependent on the temperature in the methanation reactor [58]. According to [112], at high temperatures (300°C) C-C bondings are broken due to the Ni-catalyst and unreactive C is covering its surface. Reactive C is hydrated (CH_4) or oxidised (CO_2). For lower temperature levels, CH_2 formation is assumed. CH_2 is considered as preliminary stage of CH_4 formation. The CH_2 can either cover the catalyst or be converted straight into CH_4.
For any temperature level hydration and decomposition of unsaturated hydrocarbons can be seen as parallel reactions to methanation. Therefore, carbon from CO/CO_2 and unsaturated hydrocarbon decomposition are struggling for the same hydrogen atoms. [112]
The message of this explanation is, when unsaturated hydrocarbons are converted at lower temperature levels (CH_2 is formed) less hydrogen is required for CH_4 formation than at higher temperature levels (C-atoms are formed). This in turn means that more hydrogen is available for CO methanation. In this context purposely just CO conversion is mentioned. In general, it is assumed that mainly CO is hydrated while the majority of CO_2 is converted into CO via the reverse water gas shift reaction (Eq.2.22). Just a small amount of CO_2 is straight converted into methane. [83]
Due to its almost isothermal behavior, the fluidized bed is more suitable than fixed bed solutions for biomass applications. When using highly sophisticated fixed bed systems to keep a low temperature level, especially the investment costs are significantly higher (Table 2.16). Despite the low temperature level, an advanced catalyst transportation into hydrogen richer areas is achieved through fluidization. There, it can regenerate much better and faster, as carbon from its surface can react with hydrogen and form methane. [92]

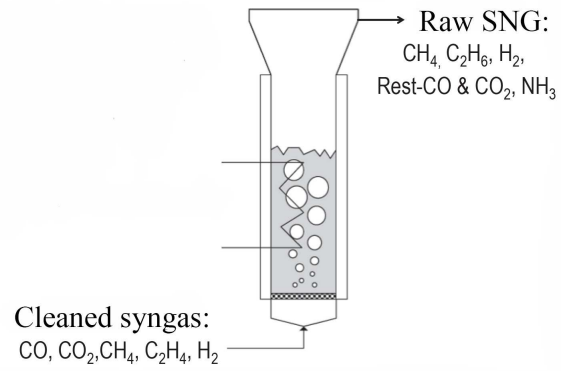


Figure 2.29: Schematic installation of fluidised bed reactor according to PSI [92]

Methanation subject to alternative reactors

The methanation systems above can be seen as state of the art technologies. However, alternative concepts are of interest to achieve further improvements in terms of long term

stability of the catalyst, heat removal and to lower the maintenance and investment costs. The most promising ones are discussed in this chapter.

	Developer	Reactor-Typ	Process-Steps	Pressure [bar]	Temperature [°C]	Year
LPM	Chem. systems	Bubble column	1	70	340	1976
BM		Fermenter		1-10	30-70	
HCR		Honeycomb		8	200-350	

Table 2.15: Overview on methanation processes subject to alternative reactors [11][59][91]

LPM is the short form of liquid phase methanation. The gaseous phase enters at the bottom of the reactor and raises upwards through a mineral oil slurry. The solid catalyst is suspended through the slurry phase. The idea is to use the high heat capacity of the slurry to provide ideal temperature conditions for methanation. A pilot plant for SNG production of 2000 m^3/h has been built and operated for long term runs. The concept did not gain the desired results due to problems considering the temperature stability of the oil. [11]
Latest attempts of 3 phase methanation are documented in the dissertation of M. Götz [43]. One of the main aspects of his work is the search for an ideal liquid phase. The following properties are considered as ideal [43]:

- *"High thermal stability and low vapor pressure to avaoid product contamination and frequent refilling of the reactor"*
- *"High solubility for the feed gases H_2 and CO/CO_2"*
- *"Increase of the volumetric mass transfer coefficient"*

According to [43], for a methanation slurry reactor, a honeycomb or fixed bed reactor should be included downstream for backup methanation. In case of an optimal operation, the slurry reactor roughly converts 85% of the fed CO_2. The residual CO_2 can be converted in an ordinary reactor type as less heat is released due to less CO_2 conversion. It is suggested that further research considering an ideal suspension liquid should be done. [43]

BM stands for biological methanation. Instead of a catalyst, micro organisms are pushing the methanation process. This can either be done In-situ parallel to biogas fermentation or in a separate reactor. For the In-situ concept, the CO_2 from fermentation is used as reactant for the methanation. Temperature conditions are set from 30 to 70°C, at a pressure level from 1 to 10 bar.[13]
The minimum load for fixed bed methanation is 10-20% of full load (for slurry of about 40%). According to [44], biological methanation can handle load changes from 100 to 0%. For Ni-based solid catalyst's the temperature has to be kept above 200°C to enable a fast re-start of the methanation unit and to avoid nickel-carbonyl formation. Further, the solid catalyst has to be separated from atmosphere as oxygen would cause oxidation/deactivation. In contrary to this, for biological methanation a re-start after 560 hours of stand-by has been performed without problems [44].

Investment costs for biological methanation are 10-100 times higher than for catalytic conversion. An indicator for higher conversion costs is the gas hourly space velocity (GHSV). It is defined as the ratio of volumetric flow rate of reactants to the reactor volume, measured at standard conditions. The GHSV describes the feed flow rate that can be fed in relation to the reactor volume each hour. The GHSV value for biological methanation is quite small in comparison to catalytic methanation. In order to achieve equal conversion rates, the reactor volume has to be higher and so are its costs.
Biological conversion is less influenced by impurities as H_2S. The harmful level of H_2S for BM is defined at 6.3 to 7.5 mmol H_2S per liter. The electricity demand is 2.5 times lower in case of catalytic methanation. The temperature level of excess heat is higher for catalytic methanation and therefore more suitable for further applications. Due to that BM has a lower overall efficiency than catalytic methanation. [13]

HCR , honey comb reactor systems promise better heat removal, less pressure drop and less catalyst abrasion. The honey comb object itself is either metallic (KIT [91]) or ceramic (MUL [18]) and coated by a ceramic washcoat. Nevertheless, the process cannot be considered as state of the art and further research is necessary to apply this concept in commercial scale even though its excellent scale up ability according to [18].

Aspect	Fluidised bed (PSI)	Cooled fixed bed (etogas)	Biological methanation (electrochaea, Microbenergy)
Temperature level, Energy recovery	+	+	-
Complexity of reactor (Upscaling)	-	+	-
Costs per unit	-	–	+
Biogas (CO_2, CH_4))	+	+	+
Woodgas (CO_2, CO, C_2H_4, CH_4)	+	-	-
State of the art	7-8	8-9	6-7
Demonstration or commercial plant (TRL)	Hüls (20 MW_{SNG}) Güssing (1 MW_{SNG})	Werlte (3 MW_{SNG})	P2G-BioCat, DK (0.5 MW_{SNG})

Table 2.16: Comparison of methanation technologies [92]

In Table 2.16, various advantages and drawbacks of the discussed methanation methods are compared. For the *State of the art* comparison 10 is the highest ranking which considers the process as commercial standard application.

2.6 Thermal storage

In the context of Power to Gas strategies is waste heat recovery an essential part to create an efficient plant. In case of thermal use of the waste heat is its storage of enormous interest. Storage systems for thermal energy are split into sensible-, latent- and thermochemical systems. Figure 2.30 shows their storage densities and level of development.

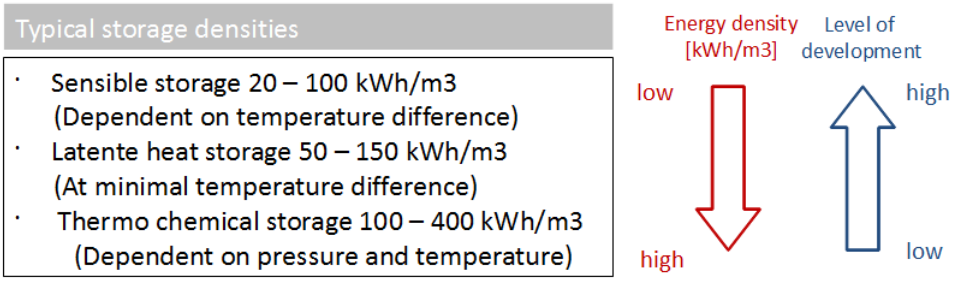


Figure 2.30: Overview on thermal storage systems [86]

2.6.1 Sensible heat storage

Sensible heat storage is the most commonly applied thermal energy storage technology. The principle is equal for all the systems. When loading the storage facility, the heat charge is raised. The heat quantity to be stored depends on the mass of storage-media, its specific heat capacity and the change in temperature difference. For discharge it is the vice versa scenario, the temperature level is lowering.
Due to a temperature gradient from the maximum at the inside of the storage tank to the minimum in its ambient, heat losses occur. Therefore, besides an optimal isolation also an optimal ratio between the storage facilities surface and volume is desirable. The load and unload -for any system- can either be done direct or indirect by implementing a separation between storage- and working media.
Sensible heat storage systems are further divided into fluid- or solid- and shortterm- or longterm-storage. Additionally, there are mixtures of fluid- and solid- or short- and longterm-storage. They are defined as hybrid concepts. Table 2.17 provides an overview on thermal storage media and their heat capacities for sensible heat storage.

Shortterm storage

Hot water storage, most commonly hot water is used for sensible heat storage. Its application is restricted by the upper temperature level of 100°C at atmospheric pressure conditions. For higher temperatures, liquid salt or solid storage are preferable. The hot water tank is isolated, it can either be operated direct or indirect with an additional heat transport media. For direct operation, different temperature levels over the tank height are created due to the density difference of warm and cold water. During the direct loading procedure, hot water is fed from the top while the same amount of cold water is removed at the tank's bottom. The floating direction is vice versa for the unload scenario. For indirect

Media	Temperature range [°C]	Spec. heat capacity [kJ/kgK]	Vol. heat capacity [kJ/(m^{3}K)]
Water	0-100	4.19	4175
Gravel, sand	0-800	0.71	1278-1420
Granite	0-800	0.75	2062
Concrete	0-500	0.88	1672-2074
Brick	0-1000	0.84	1176-1596
Iron	0-800	0.47	3655
Thermal oil	0-400	1.6-1.8	1360-1620
Gravel-water fill (37 Vol.% water)	0-100	1.32	2904
Liquid salt (e.g. 53KNO_3+40 $NaNO_2$+7$NaNO_3$)	150-450	1.3	1725-1970

Table 2.17: Comparison of heat capacities of storage media [86]

storage, the storage media remains inside the tank all the time. Its temperature is raised via heat transfer from a heat exchanger.
Due to the temperature restriction of 100°C, hot water storage is commonly applied to provide hot heating water to houses in various scales. The storage facility acts as a buffer of excess heat. The heat can be generated by any primary energy carrier of any scale. To meet the demand of peak loads, additional electric heating is sometimes installed inside a tank. [86]

Steam storage facilities consist of one or more pressurized steel-tanks. They are filled of about 90% of the volume with water at boiling temperature. While loading the generated waste heat steam is fed into the vessel and condenses in the water. The water temperature and pressure inside the vessel are rising in order to keep the water temperature constantly at boiling temperature. Due to pressure levels up to 10 bar [98] the water temperature can be above 100°C. In case of energy release a pressure drop is reduced -e.g. by opening a valve- which indicates steam generation until the temperature decreases down to boiling temperature at the new pressure level.
The stored energy is relatively quickly available. Therefore, the technology is established for years in food industry where fluctuating peak loads have to be met. The technology is not suitable for longterm storage as steam storage cannot provide a constant pressure and temperature level. [86]

Thermal oil storage is similar to the hot water storage and just applicable for short-term storage as well. Different raw oil distillates or synthetic thermal oils are applied, which enables storage temperatures up to 400°C.
The higher temperature level allows heat supply for industrial applications like process steam generation additionally to traditional heat supply for buildings. For example, process steam of about 160°C can be created at a pressure level of 6 bar. This makes the technology attractive for pulp and paper-, food- and wood-industry.

The steam inlet-temperature range is up to 400°C. Charge and discharge are usually carried out by a heat exchanger so that the storage thermal oil remains within the tank. Attention has to be brought to micro cracks in the heat exchanger that could possible transport thermal oil to the working media (e.g. water, flue gas). [6]

Liquid salt is in comparison to thermal oil operated at even higher temperatures (150-550°C inlet temperature). For heat transport mainly eutectic melting mixtures of two or more salts are applied. One of the main characteristics of an eutectic mixture is that due to the mixtures special compositions all different components are melting at the same temperature. Further characteristics of the mixture are a lower melting point and a higher heat transfer capacity, compared to pure salts. The advantages of liquid salt as storage media are the low price and long-term stability as well as its low viscosity, high heat capacity and its good heat transfer properties at atmospheric pressure. Typical storage mixtures are presented in Table 2.18. Their usage is restricted to 550°C as thermal decomposition starts at this temperature. In any case the temperature of the liquid salt has to stay above the

Salt mixture	**Melting temperature**	**Bulk weight**	**Spec. weight solid salt**	**Heat capacity**	**Temperature range**
	[°C]	**[kg/m^3]**	**[kg/m^3]**	**[kJ/kgK)**	**[°C]**
$NaNO_2$/ $NaNO_3$/ KNO_3	142	1200	2100	1.56	200-500
$NaNO_2$/ KNO_3	141	1200	2050	1.52	200-500

Table 2.18: Comparison of salt mixtures for thermal energy storage [86]

melting point. If the salt turns solid all the pipes are out of operation. Therefore, the time period for storage is restricted.
This technology was originally designed for heat buffering of concentrated solar power plants (solar tower, parabolic reflectors, ...) to cover the power gap during the night and hours of shadow. The proposed plant scale is up to 150 MW of heat storage. [6] [86]

Solid storage is either realized in a compact structure (e.g. concrete) or a bulk (e.g. gravel bulk). For the compact system a heat exchanger is embedded in the solid structure. To create an optimal heat exchange, the transfer interface has to be sufficient in size. Inside the solid phase -of storage material- the heat transport occurs by heat conduction. The heat carrier inside the heat exchanger tubes is either gaseous or liquid.
In case of bulk storage the heat transfer interface is quite large. The fluid is streaming through a disordered net of channels. Due to continuous changes of the heat carrier's floating direction and velocity the heat transfer is enhanced as well.
Bulk storage is rather applied for storage of small energy amounts than compact storage because of problems considering the realization of the heat exchanger design for compact systems.

	Alcoa-balls	Duranit-balls	Eifellava-split	Basalt-split	Diabas-split
	Alcoa 3/16”	Duranit-inter 1/4”	Eifellava 6/8	Basalt 5/8	Diabas 5/8
Particle diameter [mm]	4.8	6.4	6.1	6.6	6.2
Bulk density [kg/m^3]	2350	1400	1120	1600	1580
Void fraction [-]	0.375-0.4	0.4-0.45	0.46-0.47	0.46	0.44
Solid density [kg/m^3]	3950	2450	2100	2950	2820
Average spec. heat capacity [J/(kgK)]	1090	840	1100	1004	980
Average heat conductivity [W/(mK)]	2.1	1.7	1.4	1.7	1.8
Heat expansion coefficient [10^{-6}/K]	7.3	4.7	5.5	9.1	9.1

Table 2.19: Data of selected bulk material for heat storage [86]

The selection of bulk material is a matter of the chosen working range, thermal- and mechanical-properties (Tab.2.19), longterm stability and material costs.
Basalt-split of Table 2.19 is the most promising storage option. It is affordable and shows good thermal- and mechanic stability.
Compact storage facilities are excellent for high-temperature applications like for concentrated solar power plants. In comparison to liquid salt systems, there is no lower temperature restriction to keep the process alive. 24 hours of solar power usage and temperature storage of above 1300°C can be realized. [86]

Longterm storage

Long time storage describes a time period of days, weeks and even months. It is of growing importance e.g. to store renewable heat energy gained in summer for the colder winter months.

Hot water storage in this term is similar to the principle of short-term storage. Tank storage in general requires high efforts considering its isolation. Alternative concepts as

storage reservoirs (e.g. artificial lakes), cavern storage or storage in vertical manholes are available. Usually the type of technology is selected depending on its economic efficiency.

Water-gravel-storage is equal to hot water storage with the addition of gravel as storage media. Gravel improves the longterm storage ability. The water-gravel mixture is put into water proofed pits. The heat transport can be done direct or in-direct. Due to the gravels' lower energy density -in comparison to hot water- the required storage volume is higher (30-50%). But it is cheaper than conventional long-term hot water storage when tanks have to be manufactured.
The concept is for example applied in Eggenstein-Leopoldshafen, Germany. Solar collectors are mounted on the school building and deliver excess heat to a storage reservoir. The maximum temperature is 80°C due to the plastic cover of the pit. During autumn and winter the school builing is heated by the storage facility down to a storage media temperature of 35°C through direct heat exchange. For lower temperatures a heat pump is installed. The schools overall heat demand can be covered by about 35-40%. [86]

Aquifer storage describes large scale storage in reservoirs. Similar to hydrogen storage the reservoirs are situated in low depths (100-500 m). For thermal heat storage, water is pumped from the reservoir to the daylight. Above ground the water is heated and afterwords injected into the aquifer again, in a different drilling than it was taken from. The reservoir is loaded up to a temperature of about 70°C and discharged at a temperature level from 65 to 30°C. Due to its huge volume and the lack of extra isolation, it takes years to create an overall temperature level of the mentioned 65°C. The aquifers heat efficiency is reported as up to 80%.
This storage approach is built for industrial excess heat storage (e.g. CHP, pulp and paper). The German Reichstag and surrounding buildings are heated by an CHP plant. The plant's excess heat is stored in an aquifer 300 m deep and discharged during the winter months. [86]

Sea/Lake-water-storage is an application just for large scale power plants. Conically shaped polymers are installed vertically into the sea or lake. The plastic tube's thickness is of about 60-80 cm and its diameter and height between 30-50 m. The water inside is separated from the surrounding area, it is heated by the power plant's excess energy. The heat reutilization is for seasonal heating during the winter months. [86]

Geothermal probe is a U-shaped heat exchanger within the soil. The heat carrier is sent into the probe on the one side, warms the soil and released at the other end. The ground surface is covered by isolation but temperature losses into other directions have to be considered as well. Therefore, this system is just applicable for soil volumes beginning from 50000 m^3, in order to create a good volume to interface ratio. The probe-feed temperature is not higher than 80°C in order to garantee a long lifetime of the plastic probe. To use the warm surrounding soil, 3-5 years are necessary. After this 3-5 years, the heat recovery rate is expected as 60-70%. [86]

2.6.2 Latent heat storage

For latent heat storage, the storage material changes its phase, just in terms of solid/solid transition it remains equal (heat release through change of crystalline structure). Hence, its short form is PCM, phase change material. While heat is fed to the PCM, its temperature does not rise until all the PCM is melted, see Figure 2.31. During the melting process, a

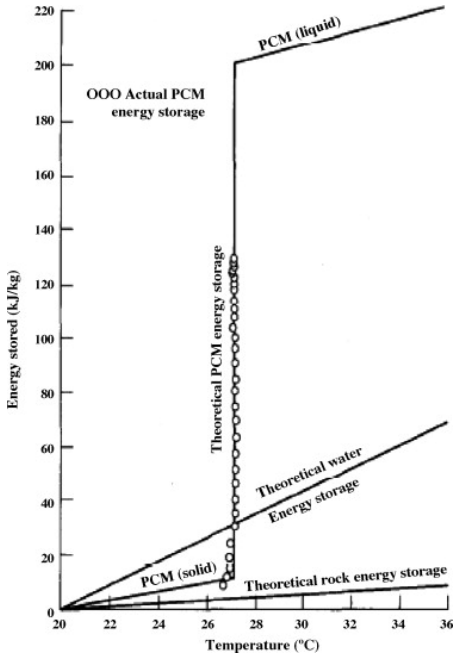


Figure 2.31: Performance comparison of phase change material (PCM), water and rock storage system [93]

disorder is created in the atom lattice which leads to a rise of entropy. For the discharge -crystallization of the storage material- the atomic order gets recreated and the entropy becomes lower. The main surplus is the constant temperature level of the storage material for charge and discharge. Another plus is its higher heat storage density and the wide range of different storage material with different melting points. However, the latent heat concept is a more expensive option in comparison to sensible heat storage.
If the heat carrier and storage media are separated by an heat exchanger, the system is no longer called indirect, it is called passive. Passive systems are having its difficulties when discharging the storage. The area close to the heat exchanger interface turns solid first. Through this layer, heat transfer to the heat carrier is restrained.
For active systems, heat exchange between heat carrier and storage phase is much better. For charging it has to be taken care that both media (carrier/storage) are not mixed together, as they have to be able to separate while discharge again.
The general aim is to create a simple system design. The choice of PCM is dependent on the desired temperature level, costs, availability, environmental impact and individual system requirements. [86]

Solid-liquid storage

The most common example for solid-liquid phase transition is water. At 0°C, a change from liquid to solid is performed for discharge and vice versa for charging. The PCM is characterized by the maximal store-able enthalpy (energy-density) and the temperature level at phase change.

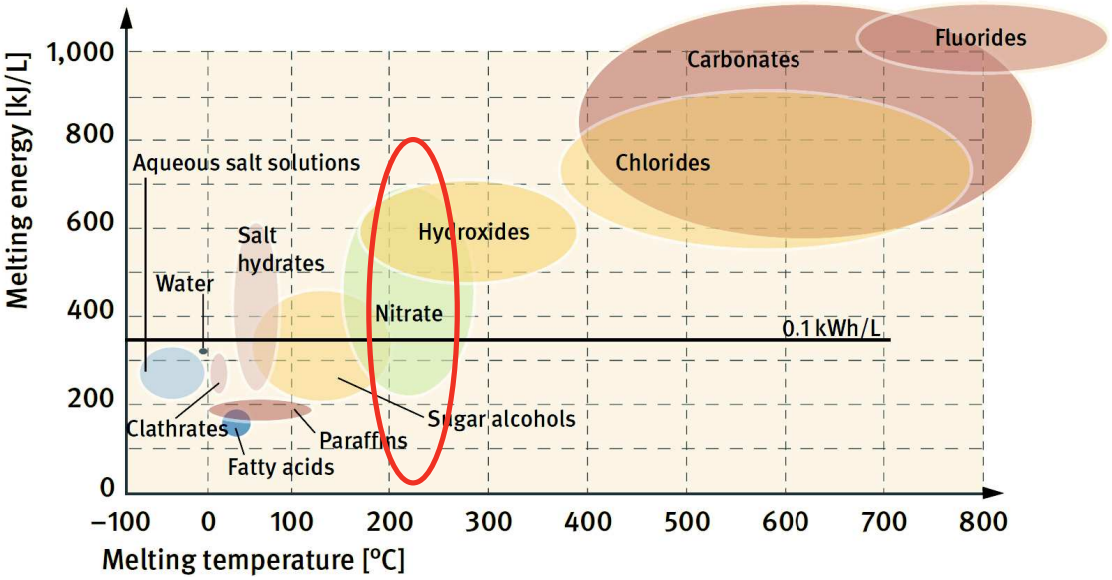


Figure 2.32: Various PCM and area of application

Figure 2.32 shows the range of application for common PCM materials. The red ellipse marks the area for solar thermal energy storage.

Solid-solid storage

A network of polyethylene does not melt while heating but its crystalline structure changes. Nevertheless, it is also considered as a PCM. Drawbacks are the material's high price and a low energy storage density of the material itself. However, on an overall perspective a low volume demand is required as no phase transition occurs. Due to that, this technology is more promising than the more volume demanding solid-gaseous and liquid-gaseous concepts. [86]

2.6.3 Thermochemical storage

It is categorized into sorption-storage and storage due to chemical bondings. Both principles are reversible. Sorption-storage is further separated into ad- and absorption storage.

Sorption storage, the principle is that a working media -e.g. water- is attracted to a solid- (adsorption) or liquid- (absorption) agent.

Solid *adsorption* agents have a porous structure and therefore a large surface. Most commonly zeolites, silica gels and metal hydrides are chosen as storage media. Their energy densities and effective temperatures differ according to their production procedure and structure. During the charging procedure, water is vaporized at the adsorbents surface. While discharge, steam at a low temperature level is fed to the system and adsorbed by the dried adsorbent agent. During the adsorption heat is released. The heat exchange between carrier and storage environment can either be operated direct or in-direct. If the storage facility is an airtight system, almost no heat losses are expected. Negative aspects at this technology are the low heat conductivity of zeolites (0.2-0.6 W/mK) and silica gels (0.14-0.2 W/mK). For in-direct heat exchange the heat exchanger -embedded in the adsorbent agent- is the only effective source to enhance the process. [86]
Table 2.20 shows data for selected adsorbents.

Energy density	200-500 kWh/m^3 (theoretically)
	130 kWh/m^3 (practically)
Temperature range	Zeolite 100-300°C
	Silica gel 40-100°C
	Metal hydride 280-500°C
Thermal conductivity	0.14-0.6 W/mK
Application area	Mainly heating of buildings in various scales
State of the art	Pilot plants, high potential
Storage volume	$\geq$ 10 m^3
Duration of storage	Shortterm, longterm potential

Table 2.20: Typical properties of adsorption agents [86]

In contrast to adsorption, *absorption* is carried out with a aqueous and strongly hygroscopic storage media e.g. lithium- or calcium-chloride. The storage- and carrier-media are mixed. Similar to the charge by adsorption, water is removed by warm air and the salt-solvent gets concentrated. For discharge the concentrated salt-solvent is attracting wet air. When water is absorbed, the air gets dry and the salt-solvent gets aqueous again.
In comparison to adsorption, the released energy is small as bondings with the liquid absorbent are weaker.
Therefore, adsorption storage is preferred for heating applications. [86]

Reversible chemical bonds, the basic principle for the reversible bonds technology is quite simple. Heat is supplied to a system for an endothermic reaction (loading). When the same reaction is proceeded in the reverse direction it is exothermic and the same heat amount is released. To achieve high storage density, high reaction enthalpies are required. Great potential for high temperature level is expected for decarboxylation of metal-carbonates and dehydration of metal-hydroxides due to their high energy densities and low material costs. Additionally, the reduction of metal-oxides is also worth mentioning.

Definition	Reaction	ΔH (1bar)	T_{eq} (1bar)	Capacity
		kJ/mol	°C	kWh/m³
Dehydration of metal-hydroxides	$Ca(OH)_2 \leftrightarrow CaO + H_2O$	112	505	364
Decarboxylation of metal-carbonates	$CaCO_3 \leftrightarrow CaO + CO_2$	167	896	113
Reduction of metal-oxides	$MnO_2 \leftrightarrow 0.5\ Mn_2O_3 + 0.25\ O_2$	42	530	136

Table 2.21: Data of selected chemical reactions for heat storage [86]

In Table 2.21 the decomposition of metal-hydroxides is shown. While heat is fed to the storage system, $Ca(OH)_2$ is decomposed into CaO and steam. During discharge, the basic compounds are forming calcium hydroxide again while the stored reaction-heat is released. [86]

2.7 Hydrogen storage

An overview for possible hydrogen storage facilities is given in this section. Hydrogen storage is a topic of interest for PtG concepts as through storage a methanation plant can be operated in times of no electricity excess. Besides a permanent SNG supply this is also beneficial as start-stop operation has harmful effects on the catalyst and other major parts. In this context stationary storage options are discussed. Hydrogen storage for fuel stations is described in detail in [35].
If geographically available, smaller amounts of hydrogen from electrolysis can also be stored in large scale facilities as caverns, which is of interest for PtG concepts.

2.7.1 Storage concepts

Hydrogen can be stored via gaseous, liquid, hybrid storage concepts and by the creation of physical or chemical bindings. The general aim is to use a high storage density concept at reasonable financial and technical efforts.

+ Storage related to phase state

Gaseous hydrogen storage

For the storage in gaseous phase, the hydrogen density is increased by compression. The applied pressure for compressed hydrogen is between 200 and 900 bar for storage in a steel and fibre tanks. [53] The compression efficiency of real gas hydrogen is of about 50% according to [35].
Figure 2.33 shows the compression work efforts as a function of the final pressure for an

ideal- and real-isotherm case, respectively an adiabatic case. For isotherm compression of real gas, the ideal gas equation was extended by an real gas factor based on empirical data. The dashed line shows the practical case of a mixture of isotherm and adiabatic multistage compression with intermediate cooling. [53]

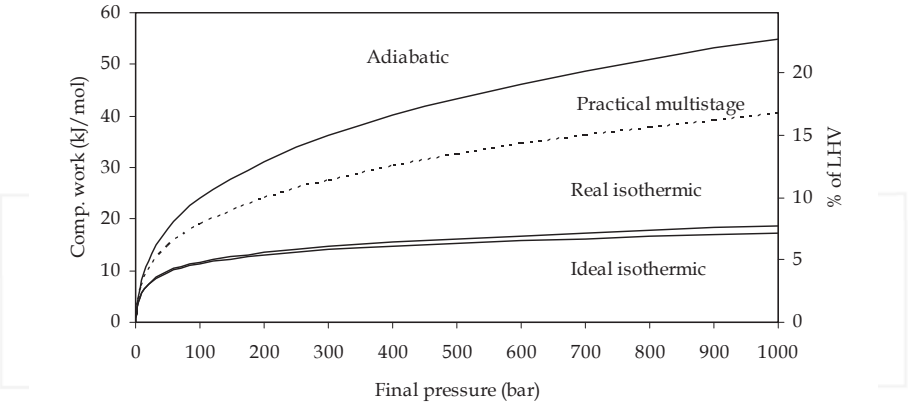


Figure 2.33: Energy demand to compress hydrogen from 1 bar to the final pressure specified on the primary axis [53]

Only a few projects are also considering oxygen storage in their publications and balances. One of them is the *ARGE Initiative Wasserstoff in Mecklenburg-Vorpommern* project where hydrogen is gained via PEM-electrolysis and further compressed to 281 bar for gaseous storage. The generated oxygen is compressed to 61 bar and stored in gaseous phase in a tank as well.[1]

Liquid hydrogen storage

In order to gain higher energy densities than for gaseous phase, hydrogen is liquefied for its storage. At ambient pressure conditions, hydrogen is condensing at -253.85°C. It is kept together by van der Waals forces. Efforts for liquefaction are relatively high. Also the complexity of storage and transportation are rising.
While the temperature for oxygen and nitrogen lowers via Joule-Thompson expansion, hydrogen has a negative Joule-Thompson coefficient at room temperature and therefore the gas would heat up instead of cooling. The practical cooling chain of moderate compressed hydrogen starts with a conventional cooling system, followed by cooling through liquid nitrogen, a mechanical expander and in the end a Joule-Thompson valve. For low temperatures the Joule-Thompson coefficient turns positive so that the hydrogen is cooled and liquefies. According to [21] the energy demand per kilogram of hydrogen is 14.2 MJ, this equals 28.4 kJ/mol H_2 and 11.7% of the LHV-based energy content. For hydrogen release -after storage- surrounding heat should be sufficient for evaporation, but due to the possibility of a freezing heat transfer media -inside the heat exchanger- an additional electric heater is sometimes installed. 892 J/mol H_2 of heat must be supplied to vaporize and discharge the stored hydrogen but due to the low boiling point of -253.85°C the heat can be taken from the ambient as already mentioned. [53]
For modern plants the overall energy demand for hydrogen liquefaction is of about 40-45% of the LHV-based energy content. According to [21], a reduction down to 25% or

even 21%([109]) of the LHV-based energy content can be expected for very large scale liquefaction. The largest plant is currently operated in the US with a production of 60 tons per day and an energy demand of 30% LHV-based. In Europe, smaller plants are in operation in France (10t/d, Air Liquide), Netherlands (5t/d Air Products) and two plants in Germany (4.4 and 5 t/d Linde). [35]

Hybrid hydrogen storage

Slush storage is a concept where hydrogen is cooled below -259°C to create a mixture of liquid and solid phase which is called slush. A share of 50 wt.% solid hydrogen results in a density of 82 kg/m^3 (Fig. 2.34). The slush density is in between the densities of solid and liquid phase. Due to its higher density it is of interest e.g. as rocket fuel. So far it is just realized in labratory scale. [35]

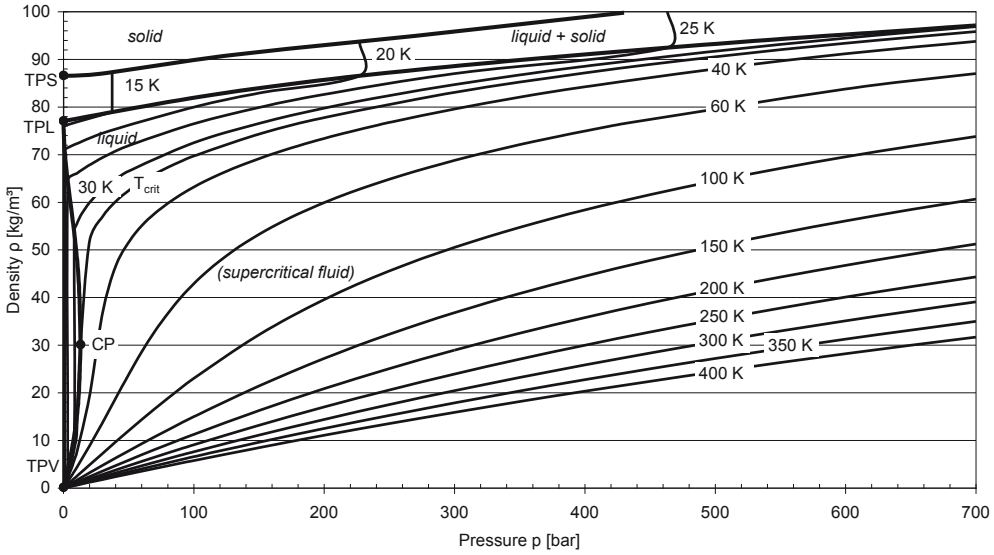


Figure 2.34: Density of hydrogen in dependency of temperature and pressure [35]

Supercritical storage, for a high density of about 80 kg/m^3 -similar to the slush case- hydrogen has to be compressed to 350 bar at 35 K (Fig. 2.34). These conditions are above the critical point, defined at 31.1 bar and 33.2 K. Due to the lack of phase transition, boil-off losses can be reduced. Because of conditions of high pressure and low temperature the requirements for the tank system are quite high. [35]

+ Physical and chemical adsorption

In case of physical adsorption molecular hydrogen and in case of chemisorption atomic hydrogen is bonded.
Interaction between a solid surface and hydrogen molecules is the priciple of physisorption.

The solid particle mainly has a porous carbon structure. The carbon grid can be formed to so called nanotubes. At temperatures of 50-80 K, storage densities of 3-5 wt.% of hydrogen have been achieved. Besides the low storage capacity also the release of hydrogen shows some difficulties. Via substitution of carbon made storage material through plastics (e.g. Polyanilin, Polypyrrol) a density increase up to 8 wt.% of hydrogen is expected. [35]

+ Chemical absorption

It appears when **Semi- and nonmetals** (3.-7. main group) form hydrogen bonds. Binding partners are separated in the category of formal positive (e.g. water, ammonia, hydrochloric acid), formal negative (silane, borane) and neutral ones also called covalent/homopolar (e.g. methane).
Hydrideions H^- are forming **saline hydrides** together with electropositive alkali- and alkaline earth metals. They are bonded through an ionic connection. Beryllium is the only species from the main group section that cannot be applied.
The most interesting storage medium for chemisorption are metal hydrides. Applied are either elementary metals (e.g. palladium, magnesium, lanthanum), intermetallic bondings, light metals (e.g. aluminum) or certain alloys (e.g. Ti-Ni-Ti_2Ni). The hydrogen atoms are integrated in the metallic structure. The main focus on research is considering intermetallic bondings. [35]

2.7.2 Large scale hydrogen storage

Large scale hydrogen storage has similar requirements as the storage of natural gas. One of the main differences is the smaller size of hydrogen molecules and therefore the tendency to diffuse easier through the storage interfaces. The five large scale storage possibilities for gaseous hydrogen are [101]:

- Tanks
- Underground storage facilities
- Salt caverns
- Pore storage facilities
- Rock caverns

Tanks

The storage of hydrogen in tanks has already been discussed in Subsection 2.7.1. Large scale concepts for hydrogen are based on the town gas era. Gasometer and telescopic gas tanks are using the same principle (volume is not constant) but both are having a low energy density due to pressure restrictions slightly above the atmospheric pressure. However, for two projects a gasometer is applied for hydrogen storage (Frankfurt-Höchst Infraserv Höchst, Gersthofen CABB GmbH). Higher pressure can be applied for spherical vessels. Its investment costs are lower due to less material costs. But still, the technology has not been used for gaseous hydrogen storage just for cryogenic approaches to store liquid

hydrogen. For tube-storage, tubes are buried in the soil like a underfloor heating system in the screed. This technology has been used in Urdorf, Switzerland. It can be used for small capacities as its material demand is relatively high in comparison to the spherical vessel type as well as due to capacity restrictions by the available area to bury the tubes. [101]

Underground storage facilities

The main advantage, in comparison to storage solutions above the ground is the cheaper availability of larger storage capacity. Along with an increase in depth of the storage facility, also the storage pressure is increasing (<200 bar).
To get an impression for the storage capacity dimensions some values are presenetd, tank storage capacities are expected far below 1 million m^3. For underground storage the German average space per storage facility is 500 millions of Nm^3.
First gas storage took place in empty oil- and gas-storage facilities and was further extended by artificially created salt caverns. Pure hydrogen is successfully stored for plenty of years in specially designed salt caverns in Great Britain and the USA. [101]

Salt caverns are, as already mentioned, man-made. At first, a hole is drilled and sealed by cemented steel-tube. Two concentric tubes are put in this hole. Through the inner one water is fed to solve salt down in the underground. The water salt mixture -called brine- is brought to day light through the cavity between outer- and inner-tube. Due to sealing reasons the small cavity between outer-tube and steel-tube is floated with nitrogen. When the cavern has reached its desired size, leaking tests have to be done before it can be filled the first time.
All the work can be carried out above-ground which makes this concept less expensive and therefore very attractive for hydrogen storage. To create a salt cavern a thickness of at least 200 meters and a typical depth of 500-2000 meters of salt-formation have to be available. Further, a certain share of insoluble components in the salt -to create stable interfaces- and the ability to utilize the produced brine have to be given. The amount of brine is assumed to be eight times the cavern volume. Opportunities for utilization of the brine are the production of salt, chlorine, sodium hydroxide, hydrogen or just send it into the sea.
A general calculation assuming a salt cavern of 500000 m^3, in a depth of 1000 m, at a pressure rate of 60-180 bar results in hydrogen storage capacity of 4900 tonnes. A typical salt cavern facility is operated for natural gas storage in Nüttermoor-Germany by the company EWE Vertrieb GmbH. 21 caverns of an average size of 400000 m^3 are in a depth of 950-1300 m. An amount of 1319 million m^3 can be stored while 1.5 million m^3 can be removed per hour. Further practical experience for hydrogen storage is given by projects of smaller amounts in the United States, especially Texas and Great Britain. Data considering leakage shows as good results similar to natural gas storage. The first salt cavern for hydrogen storage in Germany should be finished for the HYPOS-project till 2020 in Sachsen-Anhalt. In order to use salt caverns as a state of the art solution -for hydrogen storage- further leakage certificates have to be compiled as the European Union's security requirements are higher compared to the rest of the world. [101]

Rock caverns are not totally sealed by its interfaces. Cracks are in the interface. To avoid gas leakage through these cracks the storage facility has to be located below groundwater level. Groundwater is penetrating from the outside through the cracks so that gas cannot stream into the other direction. The water is gathered at the bottom of the reservoir and if required drained.
To use this technology for high gas pressure, the storage facility has to be in very low depth which is quite expensive if no mining facilities are already existing. A possibility to use or create caverns in low depths is the lined rock cavern concept where the interfaces are sealed by steel-linings. Two lined rock caverns are operated, one in Switzerland and one in Sweden. This concept is very cost intensive. Experience with hydrogen storage is not available, by now. [101]

Pore storage facilities, are not one big cavity. It is a porous system like a sponge that is covered at the top by gas dense rock layers. Due to the porous structure several in- and outlet drillings have to be made, also inside the structure horizontal drillings are made to increase the velocity for charge and discharge of the storage facility.
For **depleted gas and oil storage facilities** the proof of leakage is already given by its history of natural gas storage. The small hydrocarbon molecules have been stored within the structure for thousands of years. The pore structure has to be deep enough to apply a pressure level that enables a direct feed in, into the pipeline. A high porosity and permeability for fast load and unload has to be given as well.
Typically, hydrocarbon deposits are emptied just 50% before they are used as storage site for external energy carriers. The energy carrier therefore gets in contact with the residual hydrocarbons. This is why oil deposits are rarely used for further storage. The efforts for cleaning before hydrogen can be injected are enormous.
The rock-matrix includes various minerals that might react with hydrogen. This could cause a lack of hydrogen and the creation of products that might reduce the floating properties within the reservoir. Operation results show that bacteria is present in the deposit which might cause the same effects as minerals.
Due to the current drawbacks there is only an interest in pore storage systems for areas where no better large storage possibilities e.g. salt caverns are available. So fare, pure hydrogen has just been stored in natural gas deposits, in demonstration scale. An example in size for natural gas storage is given by the storage facility in Bierwang, operated by the E.ON Storage GmbH (depth 1560 m, gas amount 1450 million m^3, unload 1.2 million m^3/h).
Aquiferformations have the same structure as depleted gas and oil storage facilities but they were filled with salty water instead of oil and gas. Also the requirements considering depth, porosity and so on are similar. In contrary to the previous case, for aquifers there is no proof of leaking for the upper stone-layer available. Efforts in this direction are required to show the usability as storage facility. For hydrogen storage minerals and bacteria could cause unwanted reactions. A typical aquifier for natural gas deposite is located in Kalle-Germany, operated by the company RWE Gasspeicher GmbH (depth 2100 m, gas amount 215 million m^3, unload 0.45 million m^3/h). Pure hydrogen has not been stored so far. Also some of the operated natural gas deposits have been shut down due to gas losses. However, research has also been done for this hydrogen storage type. [101]

Comparison of large scale solutions , Table 2.23 shows the currently applied storage capacity for natural gas storage.
For the Middle-European area, pore storage is still a popular topic of research as the expected huge demand for large scale hydrogen to be stored can just be met when pore structure storage is included.

	Depleted gas and oil storage facilities	Aquiferformations	Salt caverns	Lined rock caverns
Security	+	+/-	++	+
Density	+	+/-	++	+
Proof of leak	+	+/-	++	+
Practical experience	+/-	+/-	++	-
Feasibility	+/-	+/-	++	+/-
Storage capacity	++	++	++	+/-
Flexibility	+/-	+/-	++	++
Contamination of H_2	-	+/-	+	++
Reactions of H_2 In-situ	+/-	-	+	++
Exploration	++	-	+	+
Cushion gas share	+/-	-	+	++
Investment	++	+/-	++	-
Operation	+	+	+	+

++ very suitable, + suitable, +/- sufficient, - less sufficient, – not sufficient

Table 2.22: Comparison of large scale solutions for hydrogen storage [101]

Table 2.22 shows that salt caverns are best for hydrogen storage, assuming a feasible disposal of the created brine.

	Pore storage facilities	Cavern storage	Unit
Amount of gas already operating	10.6	12.1	Billions of Nm^3
Amount of gas to be planed or built	1.0	7.2	Billions of Nm^3
Amount of gas Sum	11.8	19.9	Billions of Nm^3
Number of depositions in operation	21	29	

++ very suitable, + suitable, +/- sufficient, - less sufficient, – not sufficient

Table 2.23: Currently applied natural gas storage capacities [101]

3 Process simulations

For process simulation the software tool IPSEpro is used. Broadly speaking, the software tool provides a graphical interface (PSE) and a model development environment (MDK) to work with self made model libraries. At the beginning of this dissertation, a model library (**BG_Lib**) already existed with several process units e.g. for biomass gasfication. However, an electrolysis, a methanation unit and some modifications were included in the model library (**BG_Lib**). A detailed description of the software tool and the model library (**BG_Lib_2016_06**) can be found in Chapter 4.
Two different Power to Gas concepts were modeled, they are described in the following two sections.

3.1 DFB-gasification in a PtG-concept

The first Power to Gas simulation considers a DFB biomass gasifier as carbon source for methanation.
An overview of the overall process of this concept is provided by Figure 3.1, whereas Figure 3.2 illustrates the detailed process chain with all process units shown. The original IPSEpro flowsheet can be found in Appendix A.

3.1.1 Model approach

Biomass is the carbon source for the simulated PtG-approach. It is gasified through the addition of steam in a dual fluidized bed (DFB)-gasifier. A more detailed description of this technology can be found in Subsection 2.3.2.
Simultaneously, water is split into hydrogen and oxygen in a PEM-electrolyser by excess electricity. This type of electrolyser has been chosen as it is ready for operation within milliseconds.

A share of oxygen is sent to the combustion chamber of the DFB-gasifier where OxyFuel combustion takes place. For the near stoichiometric OxyFuel process, the flue gas mainly consists of carbon dioxide and a rest of water and CO due to incomplete combustion conditions. Therefore, all the flue gas can be utilized for CO/CO_2-methanation. Additionally, all the gasifier product gas is used as methanation feed gas as well.
Before the gasifier's product and flue gas are mixed, ash is separated from the flue gas by a fabric filter (see Figure 3.2). The product gas is purified from char, bed material and other particles by a fabric filter too and further treated in a Rapeseed Methyl Ester (RME) scrubber to remove all tar components. A positive side-effect of the RME-treatment is the simultaneous reduction of the water content due to condensation.

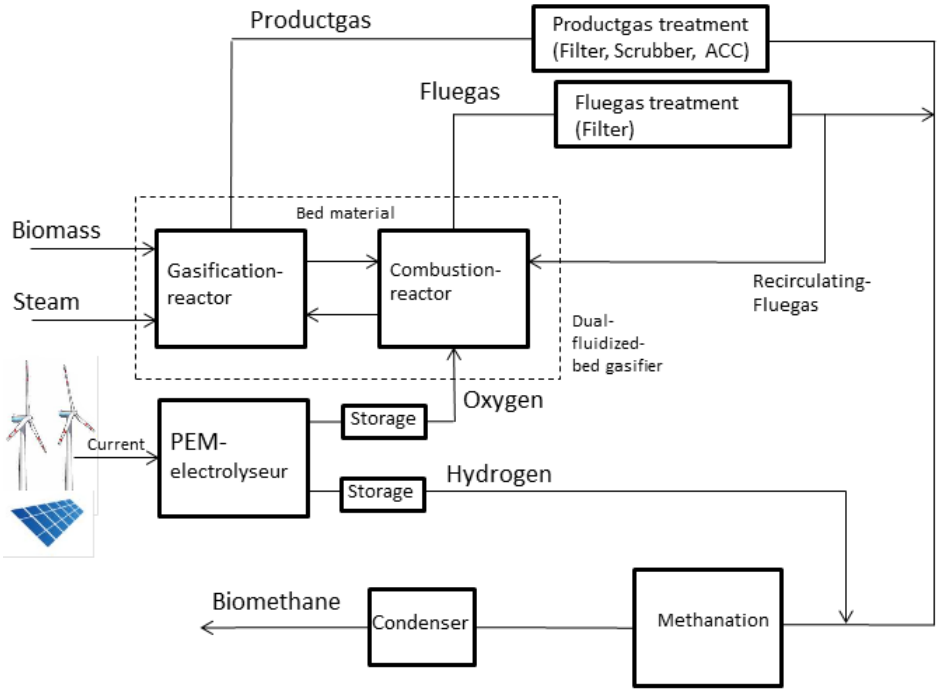


Figure 3.1: SNG-production process overview

Before the mixing of product- and flue-gas, sulphur components (H_2S, COS, Mercaptans) are removed from the product gas by an activated char coal (ACC) filter [41] in order to protect the methanation-catalyst.

The desired methanation temperature level is provided through compression as high pressure is beneficial for SNG conversion. The availability of excess electricity makes this idea sustainable.

Further, all purified gas streams are fed to the methanation reactor for SNG production. Before the injection of SNG to the natural gas grid, water is removed in a condenser. All the main process units are described in Chapter 2.

A surplus of heat is generated by the overall process, which is intended for district heating usage and steam generation for gasification.
Data are included in the simulation for possible storage of unused hydrogen and oxygen. The simulated overall power plant approach ensures 100% carbon conversion to SNG.

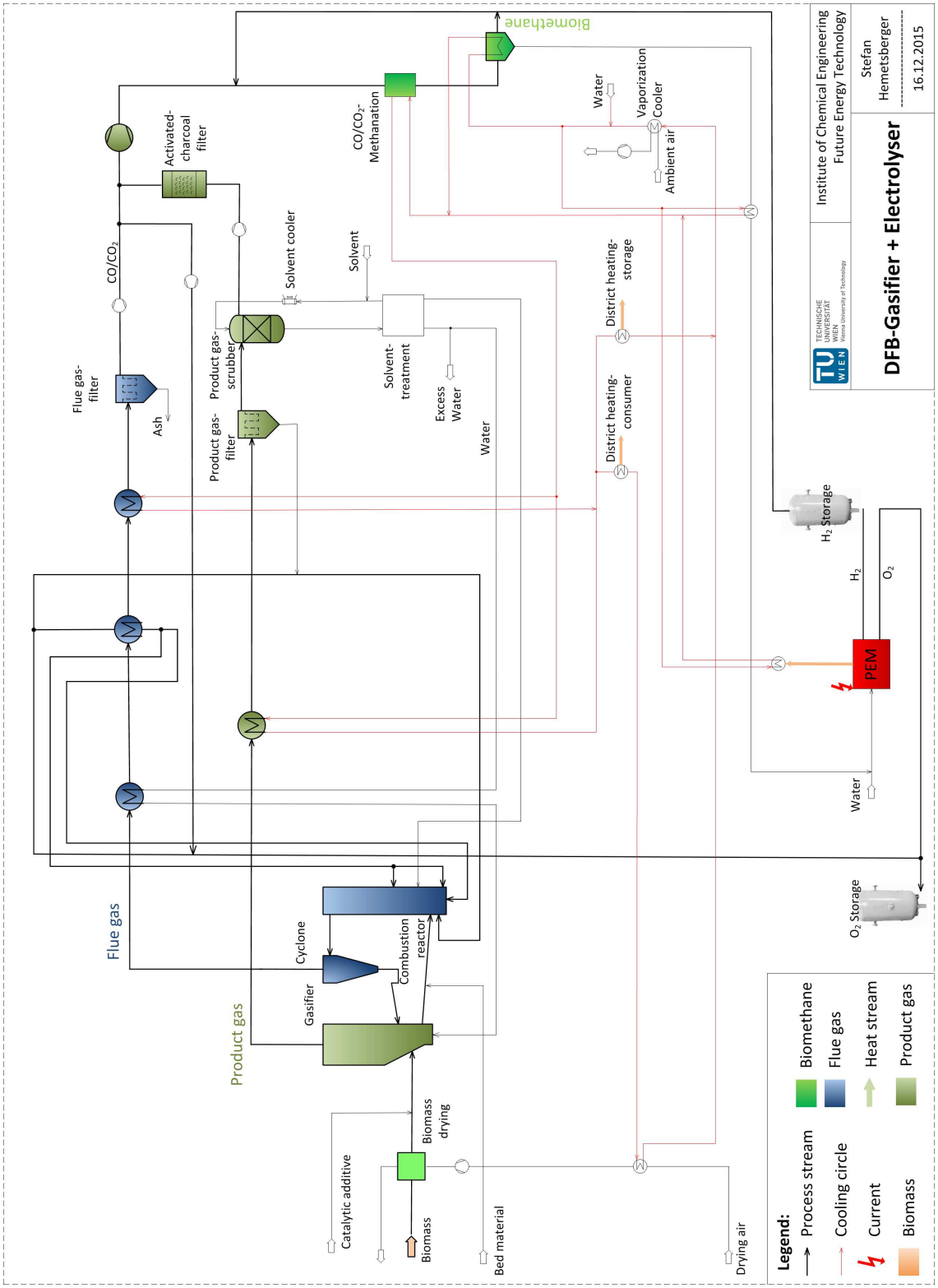


Figure 3.2: Process flow scheme of SNG-production

3.1.2 Methodology and process description

For reasons of simplicity, in the following this PtG-concept is separated into single subsections. The definition up- and down-stream is related to the methanation reactor.
The assumed efficiency coefficients of pumps and compressors ($\eta_{isentropic}$=70%, $\eta_{mechanic}$=98%) as well as electric motors (($\eta_{electric}$=96%, $\eta_{mechanic}$=98%) have its scope of application in each of these sections. Also the definition of the ambient conditions, namely 300 meters of altitude, 13°C ambient temperature and a relative humidity of 80%, are valid for the whole simulation. This conditions have been chosen due to actual values for Oberwart in the middle of October, calendar week 42.
All the values in the following subsections are setting values in the simulation.

Drying

To increase the power plant's overall efficiency, biomass drying is mandatory. The following values for biomass drying are taken from the Oberwart power plant, calendar week 45 in 2012. The water content of wet biomass is set to 34.1 wt.%. Its water and ash content (proximate analysis) as well as its ultimate analysis (C, H, N, O, N, S, Cl) is shown in Table 3.1. The biomass massflow is set to 2667.7 kg/h, no other inorganic components than ash are assumed in the feedstock.
Heat loss, drying air and biomass pressure drop are neglected.

	Variable	Setting value	Unit
Wet biomass			
	Temperature	13	°C
	Pressure	0.978	bar
	Biomass massflow	2667.7	kg/h
	Water content	34.1	wt.%
	Ash content (wf)	0.98	wt.%
	C content (wf)	49.35	wt.%
	H content (wf)	6.03	wt.%
	O content (wf)	43.5	wt.%
	N content (wf)	0.17	wt.%
	S content (wf)	0.13	wt.%
	Cl content (wf)	0.001	wt.%
	No inorganic components present except ash	100% Ash	
	Biomass massflow	2667.7	kg/h
Hot drying air			
	Temperature	72	°C
Cold drying air			
	Temperature	40	°C
Dry biomass			
	Water content	19.4	wt.%

Table 3.1: Summary of setting values for biomass drying

Ambient air is heated to a set value of 72°C and brought in contact with biomass for drying. For the drying unit no heat losses and pressure drops are assumed, neither for biomass nor for the drying agent.
The cooled drying air leaves the gasifier at a pre-defined temperature of 40°C.
The water content of dry biomass is 19.4 wt.%.

Gasification

The IPSEpro simulation for the gasifier is based on an already existing IPSEpro simulation of the CHP plant Oberwart. The related biomass settings can be seen in the previous Subsection.
The hot bed material temperature is defined as 881°C. The ultimate analysis of the containing char can be see in Table 3.2. The steam quantity for gasification is set through the mass flow of 607.3 kg/h at a temperature of 399.4°C.
The exiting bed material it is defined as pure olivine.

	Variable	Setting value	Unit
Hot bed material			
	Temperature	881.4	°C
Steam			
	Temperature	399.4	°C
	Massflow	607.3	kg/h
Gasifier			
	Heat loss	70	kW
	Fraction of fly ash	0.8	kg/kg
Product gas (PG)			
	Temperature	858	°C
	Pressure	0.9715	bar
	Dust content	15.6	g/Nm^3
	Char content	19.8	g/Nm^3
	Tar content	4.6	g/Nm^3
	HCN content (wf)	0.0	vol.%
	N_2O content (wf)	0.0	vol.%
	NO content (wf)	0.0	vol.%
	O_2 content (wf)	0.1	vol.%
	SO_2 content (wf)	0.0	vol.%
	Tar composition in PG, C	93.8	wt.%
	Tar composition in PG, H	6.1	wt.%
	Tar composition in PG, O	0.05	wt.%
Cold bed material			
	No inorganic components present except olivine	100% olivine	

Table 3.2: Summary of setting values for biomass gasification

The gasifier's heat loss is 70 kW. In the gasifier unit a fly ash fraction of 0.8 is defined for the product gas flow.
Further definitions for the product gas are made for the temperature, pressure, content of solid components, the share of five species and the ultimate composition of the containing tar. Tar in the product gas is assumed as nitrogen, sulphur and chlorine free. Most of this can be seen in Table 3.2.

	Variable	Setting value	Unit
Particle recirculation			
	Standard volume flow	1187.7	Nm^3/h
Cold bed material			
	see Table 3.2		
Secondary fluidization			
	Standard volume flow	1962.1	Nm^3/h
	Pressure	1.24	bar
	Temperature	381.2	°C
Organic scrubbing agent			
	Standard volume flow	1826	Nm^3/h
PG recirculation			
	Standard volume flow	0.0	Nm^3/h
Primary fluidization			
	Standard volume flow	492.1	Nm^3/h
	Pressure	1.24	bar
	Temperature	381.2	°C
Bottom fluidization			
	Temperature	381.2	°C
Combustion chamber			
	dp Between bottom fluid and FG	0.0	bar
	dp between particle recirculation stream and FG	0.297	bar
	dp Between prim. fluid and FG	0.217	bar
	dp between sec. fluid and FG	0.217	bar
	dp between PG recirculation and FG	0.07	bar
	Heat loss	140	kW
	Riser cross-section area	0.672	m^2
Flue gas (FG)			
	Temperature	949.9	°C
	Pressure	0.98	bar
	CO content (wf)	0.9	vol.%
	O2 content (wf)	0.0	vol.%

Table 3.3: Summary of setting values for the combustion reactor

For the OxyFuel procedure in the combustion chamber at most the stoichiometric amount of oxygen should be fed as oxygen has harmful effects on the methanation catalyst. Therefore, CO production through incomplete combustion -caused by an under-stoichiometric oxygen delivery- is preferred against excess oxygen in the flue gas. In practice, the fed amount of oxygen is calculated through IPSEpro by setting the oxygen content of the flue gas stream to zero while the carbon monoxide content is set to 0.9 vol.%. Also, the flue gas' temperature and pressure are set.
Besides the right oxygen amount, adequate fluidization velocity in the combustion reactor has to be adjusted. All standard volume flow rates (primary air, secondary air,...) are equal to the approved air fluidization of the Oberwart gasifier simulation.

The difference between the required amount of fluidization agent and the smaller amount of oxygen -for under-stoichiometric combustion- is compensated by recirculating flue gas.

The stream defined as *Particle recirculation* (Tab. 3.3) transports a mixture of recycled flue gas, oxygen and particles separated in the product gas filter. All entrances for fluidization are injecting a mixture of flue gas and oxygen. The defined standard volume flows, temperatures and pressures of the respective entrance can be found in Table 3.3. For common combustion with air, recirculating product gas is used as additional fuel. In case of Oxy-Fuel combustion no additional fuel is necessary and therefore the product gas recycling stream is set to zero for this simulation approach.
For the combustion unit the cross-section area, heat loss and various pressure drops are defined as well (Tab. 3.3).

Upstream gas treatment and electrolysis

After the gasifier the product gas temperature is lowered to 137.9°C before the fabric filter. The filter achieves a dust and char separation efficiency of almost 100% and a tar separation efficiency of about 13%, according to the given Oberwart simulation of 2012's calendar week 45.
No pressure drops are assumed for the gas stream and the removed solids.
The dust, char and tar content after the filter (Tab. 3.4) are defined via setting their content in the drain gas stream. The not-recirculated flue gas is cooled to 145.7°C and fed to a

	Variable	Setting value	Unit
PG to filter			
	Temperature	137.9	°C
Filter			
	Temperature difference gas-solid	0.0	°C
	dp gas	0.0	bar
Purified PG			
	Temperature	137.9	°C
	Dust content	0.01	g/Nm^3
	Char content	0.004	g/Nm^3
	Tar content	4.0	g/Nm^3

Table 3.4: Summary of setting values for product gas (PG) filter

fabric filter, too. For the filter a dust separation efficiency of 100% is assumed. Removal of other impurities is not considered as the flue gas stream's only solid particles are dust, see Table 3.5.
All the other settings are done in a similar way to the product gas filter.

Residual tar downstream the product gas filter is assumed to be completely removed by the RME-scrubber. Besides tar removal, 668 kg/h of water are condensing while 20 kg/h of

	Variable	Setting value	Unit
FG to filter			
	Temperature	145.7	°C
Filter			
	Temperature difference gas-solid	0.0	°C
	dp gas	0.007	bar
	Separation efficiency of solids	100.0	%
Purified FG			
	Temperature	145.7	°C
	Pressure	0.9317	bar

Table 3.5: Summary of setting values for flue gas (FG) filter

fresh RME have to be added. The product gas stream leaves the scrubber at a temperature of 36°C.
All the relevant settings for product gas scrubbing are summarized in Table 3.6. Nitrogen and oxygen in the PG results from air leakage, flushing of reactor components with nitrogen and oxygen transport from the combustion reactor.

	Variable	Setting value	Unit
Washing agent to scrubber			
	Temperature	34	°C
	Pressure	2.416	bar
	Density	882	kg/m^3
	Water content	0.0001	kg/kg_total
	Mass flow	20	kg/h
	H content (wf)	6.03	wt.%
	O content (wf)	43.5	wt.%
	N content (wf)	0.17	wt.%
	S content (wf)	0.13	wt.%
	Cl content (wf)	0.001	wt.%
Scrubber			
	Heat loss	0.0	kW
	dp between gas and solvent in the column's bottom part	0.0	bar
Purified PG			
	Temperature	36.3	°C
	Pressure	0.935	bar
	Tar content	0.0	g/Nm^3
	Char content	0.004	g/Nm^3
	C_2H_4 content (wf)	2.2	vol.%
	C_2H_6 content (wf)	0.2	vol.%
	C_3H_8 content (wf)	0.5	vol.%
	CH_4 content (wf)	9.8	vol.%
	CO content (wf)	25.6	vol.%
	N_2 content (wf)	0.18	vol.%
	O_2 content (wf)	0.1	vol.%

Table 3.6: Summary of setting values for PG scrubber

Before the activated charcoal filter the PG pressure level is raised to 1.1 bar.
To remove harmful sulfuric components, an activated char coal filter is installed. It is assumed that all the sulfur appears as hydrogen sulfide which is completely removed by the activated char coal filter.

According to [77], the product gas ammonia content is reduced by roughly 50% in the RME-scrubber. In [33] the ability of ammonia adsorption through activated char coal is described.[28] Measured data for the DFB product gas already shows a low ammonia content of 1100-1700 vppm (Table 2.4) by default. For the Oberwart gasifier simulation the ammonia content is negligible for the product and flue gas.
Therefore, in contrary to other SNG production studies (Section 1.2) the PtG-concept of this Section does not contain a separate ammonia removal unit.
For the mixing of treated product and flue gas no pressure drop is assumed.

Before the purified mixture of product and flue gas enters the methanation reactor it is compressed to a certain level via setting the methanation feed temperature -right before the methanation reactor- to 200°C.
The temperature level has been chosen due to its effect on methanation, investigated in the course of a sensitivity analysis for the IPSEpro methanation reactor source code which is documentation by Figure 4.7. It indicates a low temperature level above 200°C for a maximum methane output. Also literature data agrees with that.
Heating through compression was selected based on pressure sensitivity analysis of the IPSEpro model (Fig.3.3) and literature research as well. Renewable excess electricity makes this approach even sustainable. Figure 3.3 presents the pressure dependency of

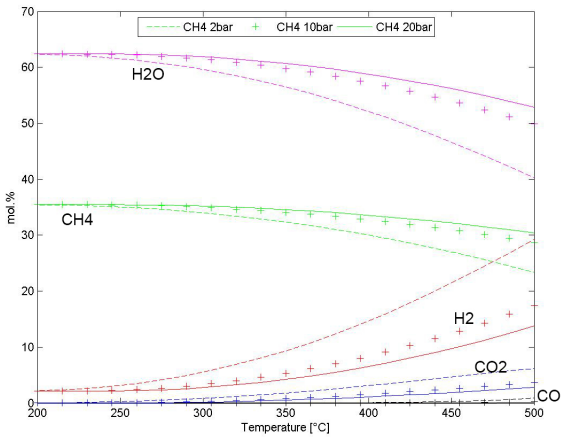


Figure 3.3: Sensitivity analysis considering methanation pressure

all major species of the methanation reactor's drain gas composition. For 2 bar of methanation pressure, lines are dashed, 10 bar dotted and 20 bar continuous. It can be ssen that low temperatures and high pressure levels are preferable for a maximum methane yield.

After the compression, hydrogen is added to achieve a stoichiometric factor of 3. The stoichiometric factor (SF) of 3 provides full carbon conversion for the methanation procedure.

In this way no further SNG upgrading but water removal is necessary.[23] In IPSEpro, the SF-ratio is set by a Free-Equation function.

$$SF = \frac{y_{H_2} - y_{CO_2}}{y_{CO_2} + y_{CO}} \tag{3.1}$$

The calculated amount of hydrogen is the starting point for the calculation of the required electric power, for the water splitting. According to literature [22], a PEM-conversion efficiency of 80% is assumed for the simulation. Based on product specifications of one of the leading PEM-electrolyser producers Proton Inc. [78] the outlet temperature of hydrogen and oxygen were set to 80°C at 30 bar. This and additional settings for the PEM-electrolyser are listed in Table 3.7.

	Variable	Setting value	Unit
Fresh water			
	Mass flow	1566.8	kg/h
	Pressure	2.7	bar
	Temperature	8	°C
Electrolyser			
	dp cooling water	1.7	bar
	Electricity conversion efficiency	80	%
Hydrogen			
	Temperature	80	°C
	Pressure	30	bar
Oxygen			
	Temperature	80	°C
	Pressure	30	bar

Table 3.7: Summary of setting values for electrolysis

Methanation and downstream treatment

For the simulated methanation, full equilibrium is assumed. The applied model of an isothermal methanation reactor represents the methanation in a fluidized bed reactor. The overall methanation procedure in the simulation model is described by the following 3 equations, for further information on the methanation procedure see Section 2.5.

$$CO + 3H_2 \rightleftharpoons CH_4 + H_2O \tag{2.20}$$

$$CO_2 + 4H_2 \rightleftharpoons CH_4 + 2H_2O \tag{2.21}$$

$$CO + H_2O \rightleftharpoons CO_2 + H_2 \tag{2.22}$$

For the reactor cooling, thermal-oil has been chosen due to the methanation's strong exotherm behavior. Heating losses, pressure drops of heating media and gas steam are neglected for the reactor. Temperature, density and water content are defined for the thermal oil feed and drain stream (Table 3.8). Isothermal conditions and no pressure drop for gas and cooling media are assumed for the methanation unit.

	Variable	Setting value	Unit
Cold cooling media			
	Temperature	87	°C
	Density	870	kg/m^3
	Water content	0.0	kg/kg_total
Methanation			
	C_2H_4 conversion	100	%
	C_2H_6 conversion	100	%
	C_3H_8 conversion	100	%
	CO methanation reaction reaches full equilibrium		
	WGS reaction reaches full equilibrium		
Warm cooling media			
	Temperature	308	°C
	Density	870	kg/m^3
	Water content	0.0	kg/kg_total

Table 3.8: Summary of setting values for methanation

Besides elementary balances, the product gas composition is created through the listed conversion assumptions and the assumption of full equilibrium for Equation 2.22 and 2.20, see Table 3.8.

In the end all the water is removed through a condenser unit by setting the drain stream's water content to zero. The condensate temperature is set to 50°C. All pressure drops are neglected for the condenser, a condensate temperature of 50 ∘C and no water in the dry SNG stream are assumed.

Storage facilities and other auxiliary components

The energy demand for hydrogen storage is calculated for different scenarios (Table 3.9). The storage is either in gaseous phase at 200 or 800 bar or in liquid phase or a metal hydride storage system is applied. The specific energy demand is provided by literature [53]. In IPSEpro the energy demand for hydrogen storage is calculated by a dataset.

Storage phase	Energy demand (LHV based)
Gaseous (200bar)	10%
Gaseous (800bar)	15.5%
Liquid	40%
Metal hydride	5-8%

Table 3.9: LHV based energy demand for hydrogen storage

Considering the energy requirements for O_2-storage there is a lack of information. The energy demand for the option of gaseous O_2-storage is calculated in IPSEpro by assuming a storage pressure of 61 bar according the ARGE Wasserstoff-Initiative-Vorpommern [1].

For this simulation, it is assumed that all the hydrogen is consumed and excess oxygen is stored in gaseous phase for any kind of further application.

To guarantee an optimal usage of the power plants thermal capacities, a district heating/-cooling system -including thermal storage- is considered in the simulation. Data for the thermal storage system is based on information of Linz AG [74]. The maximum outlet temperature of the storage tank is 97°C and the return flow temperature is 65°C. The respective temperatures for district heating are defined as 130°C supply temperature and 60°C return temperature. All the mentioned values are related to a release of LINZ AG.[74]

Of interest as well is the evaporation cooler. Evaporative cooling is applied to reduce the temperatrue level of a cooling water circle down to 20°C while the cooling air is heated to 27.1°C when leaving the unit. This option has been chosen to lower the water demand in comparison to once-through cooling systems. According to [12], evaporation cooling also requires less space than conventional cooling.
Another reason for the utilization of an evaporation cooler is more flexibility considering the power plant's location. For conventional heat exchanger cooling a river is required nearby in order to meet the desired water quantities. In case of evaporation cooling, approximately just 3% of the cooling water is vaporized and has to be recovered by fresh water. [103]

3.1.3 Pinch analysis

Plenty of heat exchangers are necessary to obtain the desired heat quantities for the respective streams. The pinch analysis is a commonly applied method to optimize the application of utility sources and to design an ideal heat exchanger network. In general, the optimization of a process results in a decrease of external energy requirements of 15-40% and 1-4 years of payback time. Further information on pinch analysis can be found in [2] , [61], [63], [69] and [36].
According to [37], the three rules of pinch technology are:

- In a heat exchanger, both streams should either be located above the pinch point or below (1. Pinch-rule)
- Above the pinch-point, no external cooling utilities should be applied (2. Pinch-rule)
- Below the pinch-point, no external heating utilities should be applied (3. Pinch-rule)

Additionally, it is mentioned that in practice 1. Pinch rule is sometimes neglected in order to reduce the number of heat exchangers. Figure 3.4 shows the temperature levels of all

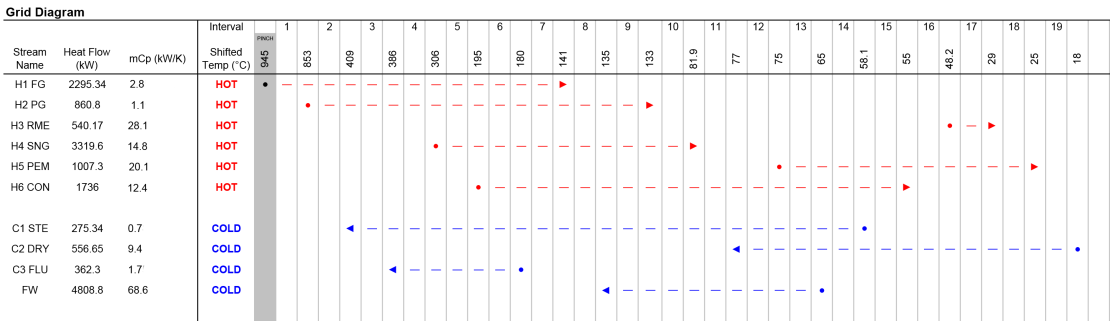


Figure 3.4: Grid diagram of temperatures

the hot and cold streams of the simulated process (Fig. 3.2). The hot and cold heat streams of Figure 3.4 are summarized to a hot and cold composite curve, shown in Figure 3.5. The main information of this figure is that all heating requirements are fulfilled by the process itself. Based on the pinch-analysis a minimum temperature difference for all heat exchangers of 108.8°C was calculated in consideration of the 2. Pinch-rule, mentioned above. The obtained pinch temperature is at 114.4°C.
The calculated minimum temperature difference and pinch temperature values are not an option for the real heat exchanger network. The system includes steam generation and temperatures of about 900°C down to about 10°C. To gain reasonable results for the minimum temperature difference and pinch temperature, the system would have to be split into a high and lower temperature area (personal information of M. Harasek).
According to [61], a minimum temperature difference of 10°C is commonly applied for chemical industry. Therefore, influenced by the results of pinch analysis but also due to consideration of practical aspects, the heat exchanger network has been designed. However, a temperature difference of 10°C has always been in place.

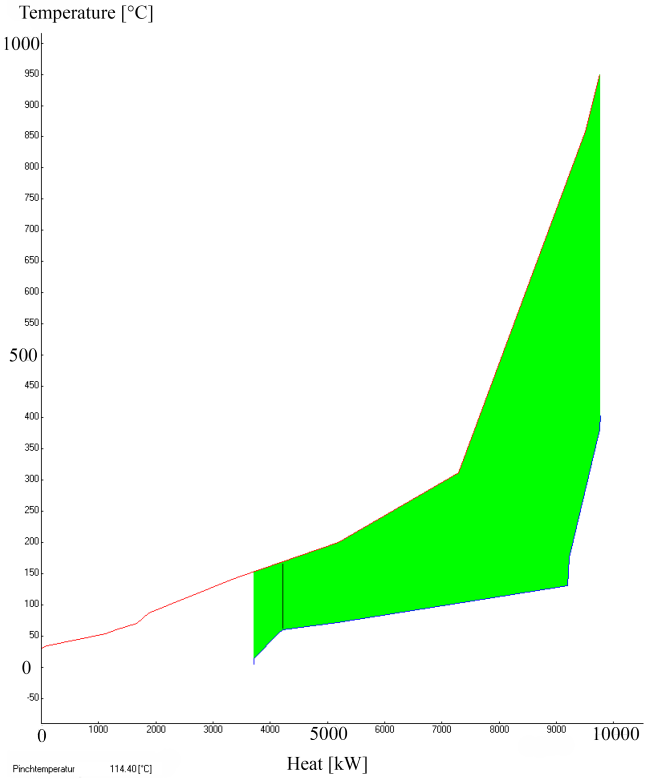


Figure 3.5: Hot and cold composite curve

3.1.4 Results

Throughout the described assumptions of the previous subsection, for each part of the plant results are gained. They are presented and discussed in the following.

To dry the biomass down to a water content of 19.4 wt.%, 33540 kg/h of ambient air is aspired and heated by 557 kW of recovery heat.

To fulfill the hydrogen requirements of 4700 Nm^3/h -at a thermal gasifier capacity of 8.6 MW- it is necessary to provide 17.6 MW of electrical energy to the PEM-electrolyser. The IPSEpro PEM-model has calculated a power consumption of 3.8 kWh/Nm^3H_2. The calculated power consumption shows good agreement with data of [22] (3.8-4.86 kWh/Nm^3H_2). The produced amount of oxygen (2350 Nm^3/h) easily meets the required standard volume flow of 750 Nm^3/h for OxyFuel combustion.

Due to fluidization of the gasifier's combustion reactor, almost 84 wt.% of its flue gas is recirculating and mixed with oxygen before entering the reactor.
The gasifier's cold gas efficiency (CGE) of 73% is defined as ratio of product gas energy to biomass (water and ash free - waf) energy content, both values are based on LHV.
The steam to fuel (waf) rate is calculated as 0.5.

Most parameters for product gas purification are set, one of the few calculated is a water content of 6.5 vol.%, downstream the scrubber unit as well as a temperature of 94.4°C, after mixing the flue and product gas streams. Due to the temperature setting, a pressure of 7.1 bar is calculated for the product- flue-gas mixture, after the compressor.
The produced amount of SNG is 1691 Nm^3/h respectively 1193 kg/h. Assuming a linear correlation between the dry biomass input and the SNG output results in a ratio of 0.55 $kg_{SNG}/kg_{dry\ biomass}$.
Different publications are having different system boundaries and constraints. Most of the time they are chosen in way to produce, so to say, not the worst results for the specific case. The calculated efficiency through Equation 3.2 is a meaningful one. Also auxiliary components e.g. blower and the energy demand unused oxygen are considered, besides the main components. For this consideration an efficiency of 58% has been calculated.

$$\eta = \frac{\dot{m}_{SNG} \cdot LHV_{SNG}}{\dot{m}_{Biomass} \cdot LHV_{Biomass} + P_{el,PEM} + P_{Pumps,Compressors}} \tag{3.2}$$

When also considering the district heating (DH) output as a product, the efficiency increases to 75%.
Hardly any other work does consider storage for excess electrolysis products. To be able to compare this results to them it has to be mentioned that without excess oxygen storage, the efficiency for SNG production rises to 59% respectively 77% when district heating is included in the balance. Table 3.10 summarizes the results in more detail.
For compensating the vaporization losses of the evaporation cooler, 3180.4 kg/h of fresh water are fed to the cooling cycle.
The recovered amount of excess heat for district heating is 4812.2 kW.
Table 3.10 shows a summary of the most relevant results of this simulation.

Biomass LHV	11141.3	kJ/kg_total
Required H_2	4661.7	Nm^3/h
Required O_2	730.8	Nm^3/h
Produced O_2	2330.9	Nm^3/h
Required electric energy	17637.5	kW
Q_trans to cooling water	1003.0	kW
DH Output	4812.2	kW
Produced SNG	1690.2	Nm^3/h
SNG LHV	49768.7	kJ/kg_total
SNG conversion efficiency	58.1	%
$SNG/biomass_{db}$	0.55	$kg_{SNG}/kg_{dry\ biomass}$
SNG+DH conversion efficiency	75.1	%

Table 3.10: Summary of simulation results

Table 3.11 presents a gas-composition overview of the most significant streams. In the second column of the Table, there are feed restriction values for the natural gas grid according to ÖVGW directive RL G31. The upper water content limit is defined through a maximum condensation point of the gas mixture at -8°C and 40 bar while for carbon monoxide no upper limit has been fixed, yet.
It can be seen that the given requirements are met by the simulation results obtained for the proposed process. Just the small amount of CO could cause some troubles. It has to be investigated if this small amount is acceptable for injecting the produced SNG into the natural gas grid.

	SNG	ÖVGW G31	Product gas gasifier	Flue gas combustion reactor
C_2H_4	0.0		1.4	0.0
C_2H_6	0.0		0.1	0.0
C_3H_8	0.0		0.3	0.0
CH_4	97.5		6.1	0.0
CO	0.04		16.0	0.7
CO_2	0.2	<2	12.5	75.8
H_2	2.1	<4	26.0	0.0
H_2O	0.0		37.4	23.5
H_2S	0.0		0.01	0.0
HCl	0.0		0.0	0.0
N_2	0.2	<5	0.1	0.0
O_2	0.0	<0.5	0.1	0.0
HHV kWh/m^3	10.8	10.7 - 15.7	-	-
d Rel. density	0.6	0.6 - 0.7	-	-
W_O kWh/m^3	14.6	13.3 - 15.7	-	-

Table 3.11: Composition of the most significant gas streams in vol.% and HHV, d, and Wobbe index for the SNG-production process

At the bottom of Table 3.11 the higher heating value (HHV) and relative density d (Eq.3.3) at standard conditions (1.013 bar, 0°C) are listed to calculate the Wobbe index (Eq.3.4).

$$d = \frac{\rho_{g,0}}{\rho_{air,0}} \tag{3.3}$$

$$W_O = \frac{HHV}{\sqrt{d}} \tag{3.4}$$

The Wobbe index is a characteristic value for the exchangeability of gases considering the thermal power and the application for different gas devices. For an equal Wobbe index, the thermal power of two different gases is equal even if their heating values are different.[72] For example, due to the addition of 20% hydrogen to a natural gas stream, the heating value is reduced by 15% as the energy content of hydrogen is just one third of the one of natural gas. Whereas, the Wobbe index is just reduced by 5% as the low hydrogen density is compensating the lower energy content. [52]

In order to verify the simulation results, Table 3.12 and 3.13 list the overall energy- and mass balance. Negative energy values are related to the energy balancing principle where sensible heat and lower heating value (LHV) are summed up. The principle is explained in the course of Figure 4.3, Subsection 4.1.3. The minor deviations for the overall balances are related to round-off errors.

Energy Balance		
INPUT		
Biomass	8.2	MW
Drying air IN	-0.1	MW
PEM P_el	17.5	MW
Fresh water evaporative cooler	-2.2	MW
Freshwater PEM	-1.1	MW
Fresh RME	0.2	MW
Electric input pumps and blower	2.4	MW
Fresh air evaporative cooler	-0.7	MW
Sum	24.2	MW
OUTPUT		
Drying air OUT	0.2	MW
Oxygen stream to storage	0.4	MW
Air OUT evaporative cooler	1.4	MW
SNG	16.5	MW
District heating	4.9	MW
Excess water RME tank	-0.2	MW
Solvent cooler	0.5	MW
Pipe losses	0.1	MW
Losses electric pumps and blower	0.3	MW
Gasifier losses	0.1	MW
Riser losses	0.1	MW
Sum	24.3	
Difference	0.1	MW

Table 3.12: Energy balance of the total power plant

Mass Balance		
INPUT		
Biomass	2667.7	kg_total/h
Drying air IN	33540.2	kg/h
Fresh RME	20.0	kg/h
Fresh water evaporative cooler	3180.3	kg/h
Freshwater PEM	1566.8	kg/h
Fresh air evaporative cooler	200398.5	kg/h
New bed material	20.0	kg/h
Catalytic bed material	20.0	kg/h
Sum	241421.3	kg/h
OUTPUT		
SNG	1193.0	kg/h
Oxygen storage	2284.3	kg/h
Drying air OUT	34024.1	kg/h
Condensate excess	283.7	kg/h
Flue gas ash	57.2	kg/h
Air OUT evaporative cooler	203578.9	kg/h
Used bed material	0.0	kg/h
Sum	241421.4	kg/h
Difference	-0.004	kg/h

Table 3.13: Mass balance of the total power plant

Sankey diagram, Figure 3.6 presents a Sankey diagram of the simulated process. All of the energy values, beginning from wet biomass to biomethane (SNG), are calculated as sum of sensible heat and the lower heating value (Q_298_and_P). The advantage of using Q_298_and_P for the energy flow balance instead of h_total is described in Subsection 4.1.3 and the dissertation of M. Stidl [102]. Q_trans represents the heat released or consumed, if heat exchange occurs.
The Sankey diagram identifies the electrical power for electrolysis as main energy source of the process. The condensate stream and the fresh water stream are connected to provide process water to the electrolyser.
The surrounding area of the gasifier has a slightly colored background. Besides the thick product gas stream in this area, the smaller streams are mainly transporting energy to the gasifier. They are described in the following at first.
For legibility reasons, the oxygen stream to the combustion chamber is not pictured. Its energy content is just 0.01 MW. The rest of the oxygen is pressurized and stored. The energy effort for compression is about 0.46 MW. This unused oxygen can eventually be sold. The hydrogen stream is completely added to the gasifier's product-flue gas mixture.
Before the biomass is gasified, it is dried. Ambient air with an energy content of -0.12 MW is aspired by a blower which is responsible for an sensible heat addition of 0.04 MW.
Several energy streams of the product- and flue-gas treatment are used to dry and preheat feed streams of the gasifier. In the *Vaporizer* unit (Q_trans 0.28 MW) the flue gas stream is cooled while the *Water* stream (-0.21 MW) is vaporized before it is sent in the gasification reactor. The oxidant mixture of recirculating *Flue gas* (0.39 MW) and oxygen from electrolysis (0.01 MW) is preheated by the heat-exchanger *Oxidant Preheater* (0.36 MW). *Loaded RME solvent* (0.29 MW) and *Particles* (0.45 MW) are returned into the combustion chamber as additional fuel. The *Loaded RME solvent* stream mainly contains used RME, condensate-water and ash. The *Heat Loss* for gasification and combustion chamber is summed up to 0.21 MW.
Water is separated from the loaded RME-solvent in a scrubber-tank. The major fraction from the scrubber-tank is recycled and used as scrubbing agent again. The revitalized solvent is declared as *Circulating RME solvent*. Due to the solvent's high chemical energy its drawing scale is different to all the other streams.
The energy increase by the *FG-Filter* unit is created due to the heat input by an electric blower of 0.01 MW which is hardly visible. That applies to the recirculating flue gas stream as well, where an energy increase of 0.06 MW is caused by a blower.
As presented in Figure 3.2 an evaporation cooler decreases the cooling cycle's temperature. The energy released to the ambient is 0.86 MW.
The energy content of the desired SNG is calculated as 16.52 MW and 4.81 MW of waste heat are fed into the district heating grid.
For some minor components the energy transport is hardly visible. But in general all components are included in this energy system consideration and its efficiency calculations.
The overall energy balance for the whole process can be seen in Table 3.12.

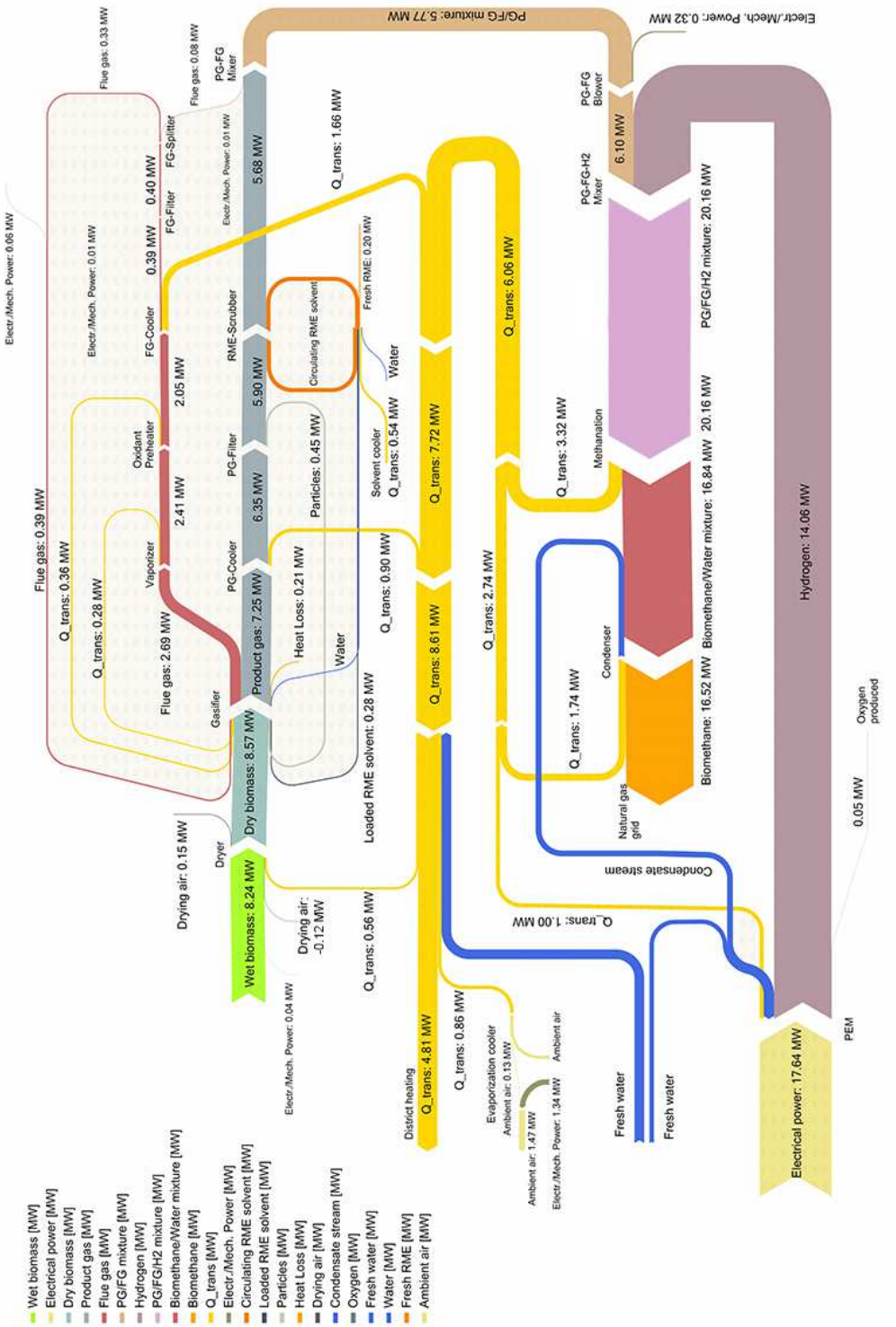


Figure 3.6: Sankey diagram of the SNG-production process

3.2 Biogas in a PtG-concept

The Power to Gas simulation of this topic was created in the course of the FFG -Austrian Research Promotion Agency- project called *Entwicklung eines katalytischen Prozeses zur Methanisierung von CO_2 aus industriellen Quellen*. The project was realized under the leadership of the institute for Industrial Environmental Protection and Process Engineering, Montan University Leoben and in collaboration with other research and industrial organizations. This simulation and recreation of a designed methanation unit and its system relevant environment was one of the work packages of the project which was carried out as part of this dissertation. The project does not consider any specific carbon source. In the course of this dissertation, the scenario of carbon dioxide from a biogas plant is considered. A biogas plant has been chosen as it is a second -besides biomass gasification- promising renewable carbon provider.

3.2.1 Model approach

As for the PtG approach of Section 3.1, water is split by excess electricity into hydrogen and oxygen in a PEM-electrolyser. The carbon source does not require any purification as the system boundaries are set in a way that assumes purified carbon dioxide. Therefore, pure hydrogen and carbon dioxide are fed for methanation. The methanation is done under adiabatic conditions in a fixed bed reactor.

One of the project aims is to show the ability of product gas upgrading by a membrane on its own. The main purpose of the membrane cleaning is the recovery of undesired hydrogen in the product gas stream of the methanation reactor.

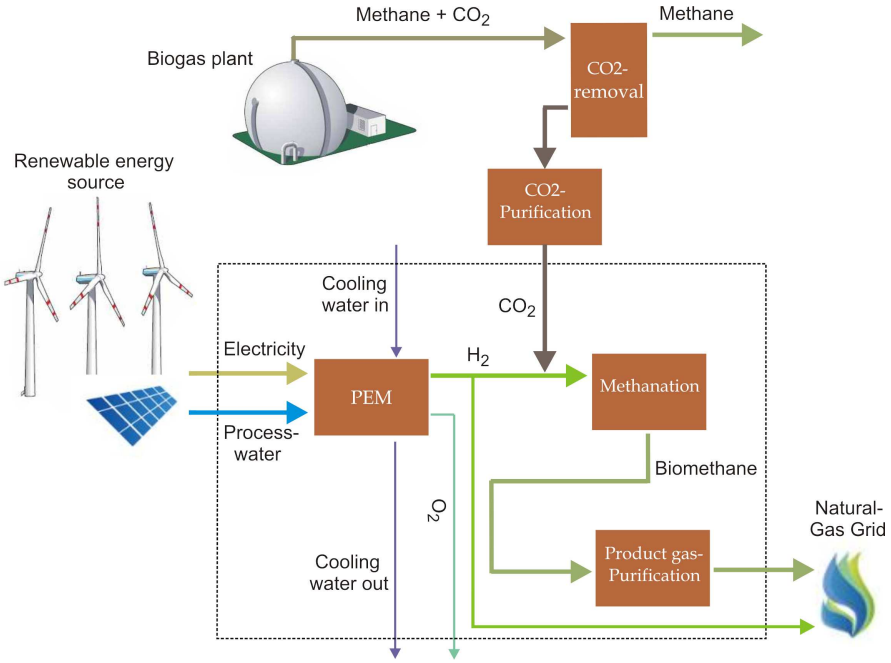


Figure 3.7: Scheme including simulation system boundaries

Figure 3.7 shows the considered simulation scenario. The dashed line shows the system boundaries of the IPSEpro simulation. The biogas plant of Figure 3.7 is not considered for the simulation but for the overall efficiency calculation. Neither hydrogen nor waste heat storage is considered for the IPSEpro simulation.

3.2.2 Methodology and process description

The original IPSEpro flowsheet can be found in Appendix A. Figure 3.8 is an illustration of the related IPSEpro simulation.

The simulation is a forecast to the laboratory fixed bed methanation experiments at Leoben University of Natural Resources and Applied Life Sciences.

Pure hydrogen and carbon dioxide are mixed with the permeate stream of a low selective membrane unit. The standard volume flow of hydrogen has been chosen as 14 Nm^3/h. The required electricity is calculated for an electrolysis efficiency of 80%. Based on product specifications of one of the leading PEM-electrolyser producers Proton Inc. [78] the outlet temperature of hydrogen and oxygen were set to 80°C at 30 bar.

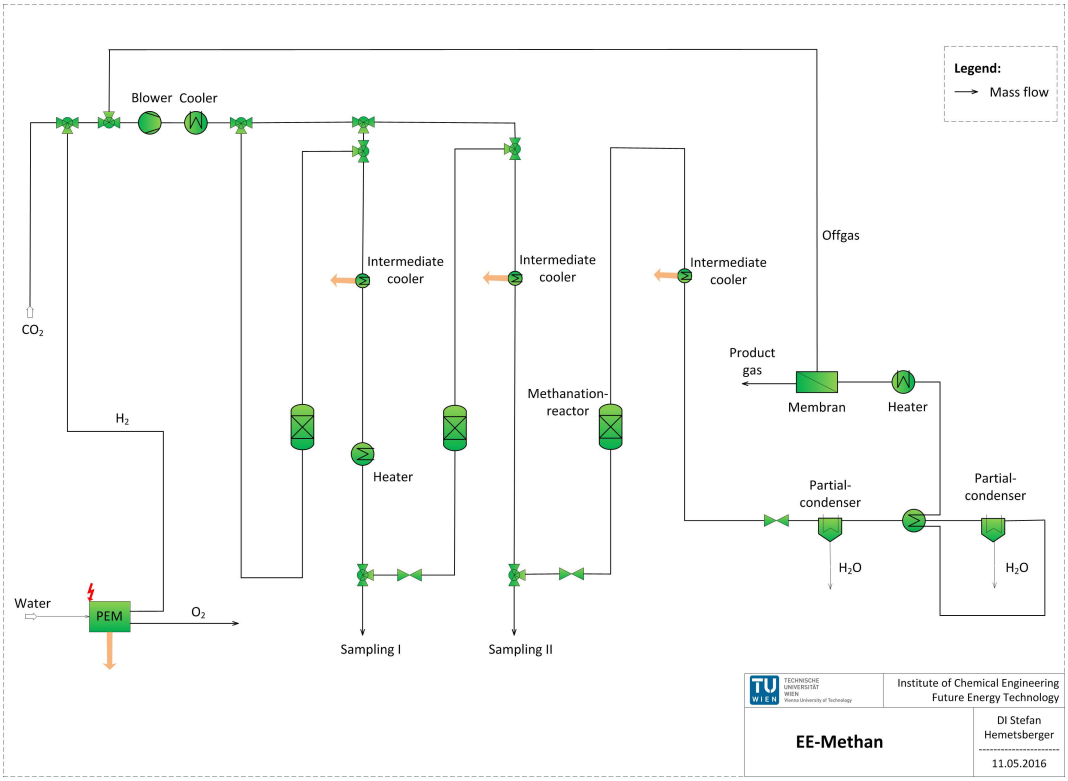


Figure 3.8: Flowsheet of the simulated process

The compression ($\eta_{isentropic}$=70%, $\eta_{mechanic}$=90%) step was implemented due to the assumptions of a pressure drop for the recycling permeate stream and a CO_2 feed pressure at atmospheric conditions. The H_2/CO_2 ratio is set to four, after the injection of the membrane's recycle stream. A cooling step downstream the compression is necessary to get the

intended temperature level of 200°C at 10 bar before the first of three methanation steps.

All the fixed bed reactors are modeled as adiabatic reactors. The rectors are modeled with the assumption of full equilibrium for the CO-methanation and WGS reaction.

For downstream product gas treatment a membrane unit is installed. It is operated at a pressure of about 9 bar. While the retentate pressure drop is neglected its permeate pressure is set to 1 bar. The permeances of Table 3.14 are taken according to [16].

Species	Permeance $Nm^3/(m_2 \cdot s \cdot bar)$
H_2	$2.53 \cdot 10_{-5}$
CH_4	$2.64 \cdot 10_{-6}$
CO	$2.97 \cdot 10_{-6}$
CO_2	$1.39 \cdot 10_{-5}$

Table 3.14: Permeances of the low selective membrane unit

For the overall efficiency of SNG production from biogas related CO_2, the biogas plant and carbon dioxide separation is considered through the following assumptions.

Variable	Value	Unit
Degree of biomass decomposition	202	m^3 biogas /ton biomass
CH_4 share in biogas	52	vol.%
CO_2 share in biogas	48	vol.%
Temperature of biogas	45°C	°C
Pressure of biogas	1.013	bar

Table 3.15: Assumptions for efficiency calculation of an biogas unit

According to [108], corn silage is the most common biomass applied for biogas production. Therefore, the degree of conversion for biomass decomposition and the biogas composition are defined for corn silage, based on [34]. The temperature and pressure conditions for fermentation are provided by [40]. These values are summarized in Table 3.15.
For the separation of CH_4 and CO_2 MEA washing is considered. Both streams are assumed as 100% pure, after their separation. [46]

3.2.3 Results

At first, results for the PtG simulation in IPSEpro are presented.
To produce the desired hydrogen standard volume flow of 14 Nm^3/h, a required electricity quantity of 52.45 kW is calculated in IPSEpro. The set H_2/CO_2 ratio is produced through addition of 3.5 Nm^3/h of carbon dioxide.

The three step methanation unit just shows a slight temperature increase for adiabatic conditions due to the small reactant mass flow. As expected, through the highest feed reactant concentration, the first reactor's heat increase (ΔT= 120°) is the highest followed by the second one (ΔT= 60°) and for the last one almost no conversion and heat increase occurs.

For SNG removal, Table 3.16 shows the calculated membrane selectivities for carbon dioxide and hydrogen related to methane, which is the substance of slower permeation that should remain in the retentate flow.

	CO_2	H_2
Selectivity	5.3	9.6

Table 3.16: Membrane selectivity related to methane (CH_4)

The selectivities are calculated according to [16], see Equation 3.5.

$$S_{ij} = \frac{P_i}{P_j} \tag{3.5}$$

The final synthetic natural gas (SNG) composition (Table 3.17) is calculated for a membrane area of 0.9 m^2.
The variable P stands for the permeance of the components i and j. Where i is the faster permeating species and j the one slower permeating.

Besides the mentioned hard facts, you can estimate that the required energy demand for heating can easily be covered by excessive process heat.
Therefore, it is not considered as an effort for the efficiency calculation (Eq. 3.6).

$$\eta = \frac{\dot{m}_{SNG} \cdot LHV_{SNG}}{P_{el,PEM} + P_{Compressor}} \tag{3.6}$$

The simulation's overall efficiency according to Equation 3.6 is 61%, without feed gas compression it is 67%.

Table 3.17 is a summary of all relevant results.

	SNG	ÖVGW G31	Gas mixture after compression	Recycle stream
CH_4	98.4		2.5	49.2
CO	0.0		0.0	0.0
CO_2	1.0	<2	19.5	8.6
H_2	0.6	<4	78.0	42.3
N_2	0.0	<5	0.0	0.0
O_2	0.000	<0.5	0.000	0.000
HHV kWh/m^3	10.9	10.7 - 15.7	-	-
d Rel. density	0.56	0.55 - 0.65	-	-
W_O kWh/m^3	14.56	13.3 - 15.7	-	-

Table 3.17: Composition of the most significant streams in vol.% and Wobbe index for the SNG-production process

A description and interpretation of the Wobbe index can be found for Table 3.11 in the previous section.

For a reasonable comparison with other relevant PtG concepts, it was necessary to include the idea of biomass conversion to gas into the system boundaries.
Based on literature [56], a biogas stream was simulated in IPSEpro to calculate a representative biogas density (1.1 m^3/kg) and lower heating value (14.2 MJ/kg). The calculated biogas unit efficiency is 0.5, based on Equation 3.7 and the assumptions of Table 3.15.

$$\eta = \frac{V_{Biogas} \cdot \rho_{Biogas} \cdot LHV_{Biogas}}{m_{Biomass} \cdot LHV_{Biomass}} \tag{3.7}$$

For the overall plant efficiency, including the biogas plant, some calculation details have to clarified.
The separated CH_4 content (3.8 Nm^3/h) of biogas to provide 3.5 Nm^3/h of CO_2 for methanation, is considered as product for the overall plant efficiency as well.
Heat recovery is not considered for the efficiency.

By considering all this points, the plant's overall efficiency is calculated as 55%.

3.3 Comparison

For a comparison of the dissertation's SNG production through biomass gasification (Section 3.1 to similar concepts (Subsection 1.2.2), all assumptions and constraints have to be considered.
In contrast to this work, for the ZSW project electricity is produced (ORC η_{el}=17%[23]) from waste heat, while for the ECN study, the MILENA gasifier unit -about 83% cold gas efficiency for a supposed biomass moisture of 25% [32]- is not included for the overall plant efficiency evaluation, see Subsection 1.2.2.
Based on a rough evaluation the work of ZSW (70-75%) and ECN (85%) show similar or even lower overall plant efficiencies as it is computed for this dissertation's Power to Gas concept of CO and CO_2 from biomass gasification. If the sum of minor auxiliary electric efforts (e.g. blower) is not considered for this simulation, its overall plant efficiency is of about 83%.
When all the auxiliary energy sources and sinks are included, the generation of SNG through biomass gasification results in an efficiency of 58%, if they and the district heating output are not, an efficiency of 64% is calculated.
According to [105] it takes 0.3 kWh of energy to produce one kilogram of oxygen through an air separation plant. If the highly pure oxygen is applied for external applications this has to be considered as well. The inclusion of the energetic foot print of the stored excessive amount of oxygen both efficiencies increase by 2%.

For the comparison of both simulated projects, the same system constraints (Table 3.19) result in an efficiency of 59% for biomass gasification (Section 3.2)and 55% for the biogas approach (Section 3.2).

Due to that figures, at first sight gasification appears as the better option for carbon supply. However, it also has to be taken into count the simplicity of the biogas approach simulation. An actual biogas stream also contains potentially harmful components for the methanation catalyst, see Table 3.18. These components have to be removed, which would cause additional, but not considered, losses.

Species	vol.%
CH_4	50-70 vol.%
CO_2	25-45 vol.%
H_2O	2-7 vol.%
H_2S	20-20000 ppmv
N_2	<2 vol.%
O_2	<2 vol.%
H_2	<1 vol.%

Table 3.18: Average biogas composition according to [56]

Also the separation of CH_4 and CO_2 e.g. through MEA washing can require additional unit components as upstream-removal of sulphur components and the installation of a downstream water-scrubber due to the MEA's high vapor pressure. [46]
Also literature does not give a more precise definition of the efficiency for biogas related

Power to Gas concept's. According to Sterner [100], the efficiency range for biogas based SNG production is defined from 57 to 68%.

As so many different system boundaries and efficiencies have been presented in the recent subsections, all of them are summarized in Table 3.19 to maintain an overview.
The last column presents the efficiency according to [100]. The line *C conversion* is for the carbon supplier, either DFB gasifier or biogas plant. As auxiliary components (*Auxiliaries*) smaller energy sinks (e.g. blower) or sources are declared. *O_2 external* considers usage of stored oxygen. For the Sterner consideration it is not defined if auxiliary components are included in the efficiency calculation.

	DFB									Biogas				Sterner
PEM		x	x	x	x	x	x	x	x	x	x		x	x
C conversion	x	x	x	x	x	x	x	x	x			x	x	x
O_2 storage		x	x	x	x									
Auxiliaries		x	x	x	x			x	x	x			x	
SNG		x	x	x	x	x	x	x	x	x	x		x	x
DH			x		x		x		x					
O_2 external		x	x		x		x		x					
%	75	60	77	58	75	64	85	**59**	77	61	67	50	**55**	57-68

Table 3.19: Summary of all efficiencies calculated

4 Modelling and process simulation

4.1 Definitions

4.1.1 Model

Almost anything can be described by a model, especially when talking about science. The model can either be simple or rather complex. The most relevant aspects for the creation of a simulation are described by T. Pröll in his PhD thesis [76][page 32], also the strategy of creating a mathematical model is listed in this context.

As this chapter is an description of the IPSEpro structure and the biomass gasifiifaction library (**BG_Lib**) as well as a guideline for future students and employees of the Future Energy Technology working group, additional information to the cited source is given in form of the referred chapters and page numbers which enables more comfortable and faster working. This additional quotations are exclusively made for this chapter.

4.1.2 General information on process simulation

The simulation of a process is the recreation of a process in a model in order to obtain knowledge about it. Due to simulation, a forecast can be given on new process designs and their behavior, but also already existing processes can be optimized. [39]
Further information e.g. on the solution strategies for process simulation in general, the simulation aims and the way to reflect the generated figures are given by [76], Subsection 3.1.3 and [102], Section 3.1. To get an first idea of the two solution strategies and their properties they are listed in Table 4.1.

Solution strategy	Advantages	Disadvantages
Sequential modular	Units can be described very detailed	Time-consuming iterations for complex flowsheets
Equation orientated	Quick convergence even for complex flowsheets when good starting values are applied	Limited degree of detail, difficult localization of errors

Table 4.1: Strategies for solving model equations [76]

For sequential modular calculations the whole simulation is split into subsets of equations where one subset -e.g. a model unit- is treated after an other in the order as they are connected in the flowsheet. The equation oriented procedure can be seen as a global approach, all equations are treated simultaneously. [15]

4.1.3 Process simulation in IPSEpro

IPSEpro simulates stationary power plant processes. It is a package of several software units. Its structure can be seen in Figure 4.1. The programs heart is its Kernel, it consisting of analyser and solver. Due to reasons of analysis, the equation orientated solution approach is applied. It separates the overall system into single pieces. The single subsystems are solved after the Newton-Raphson procedure.[76] The Newton-Raphson procedure is described for one and two dimensional use -including examples of application- in the dissertation of M. Stidl [102], Subsection 3.2.4. The Kernel is embedded in the Process Simulation Environment (PSE).
PSE provides a graphic interface to the user and is one of the two major software parts that are presented in this chapter. Flowsheets are created by connecting several process components with each other. The various shells around the Kernel of PSE are showing the adaptability of the simulation environment. The commercial available model library contains models for standard process components and can be extended by any user specific input.
All the models are collected in one model library. Each model library is saved as a file in the format of **.IML** (IntelliJ IDEA Module). The IML-file is created by compiling the model library's source code. The file containing the library's source code can be edited in the model editor. Both together, editor and compiler, are the second major software part of IPSEpro which is also described in this chapter, it is named Model Development Kit (MDK). The file-format for the model library source code is named after the software tool, **.mdk**.
Some library components are asking for physical property data. This data is provided by property library files. They are having the format **.DLL** (Dynamic Linked Library) and they are generated by compiling a source code written in C or C++.

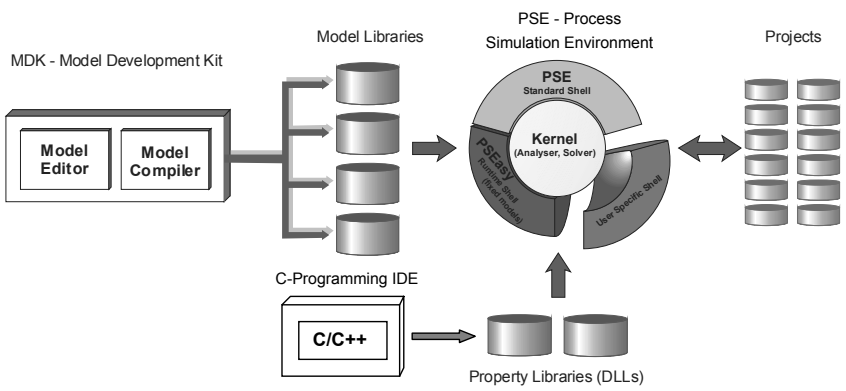


Figure 4.1: Description of IPSE software package [94]

The IPSEpro program structure is also described in [94], for a German version see [76][3.2.1], [102][3.2.2], [60][3.1.2]. All of the four descriptions are quite similar but still have its slightly different focus.
Helpful information on the work in PSE, how to define a flowsheet best and other advises are given by M. Stidl [102], Section 3.8.

Library components

In the previous subsection the three different classes of models are summarized as library components. They are also known as model classes. The following definitions of them are according to [76].

Connections are transferring information from one unit to another. A connection describes either a mass flow or mechanical shaft. Variables of any global type -see below- can be used within a connection. Within the connections, thermodynamic equations of state are implemented. All applied thermodynamic equations and variables are very detailed described in [102], Section 3.3. The respective generation of the physical property data for the thermodynamic equations and variables is described in German in [102] Subsection 3.4.2. In [76] -Section 4.3- all the relevant thermodynamic equations and the related physical property data generation is summarized. The description of the lower (LHV) and higher heating value (HHV) is its own section (4.2) in this work. The information of [76] is also available in abbreviated form in English language [104].
Figure 4.2 pictures the hierarchy of the various connection classes. In the second highest

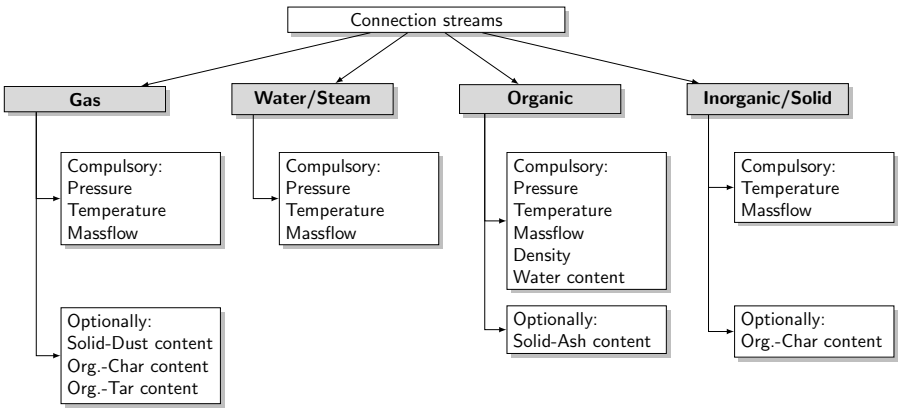


Figure 4.2: Variables of connection streams to be defined [102]

level of Figure 4.2 the considered kind of mass flow is declared. The level below, the listed variables have to be defined in order to make the stream converge. The lowest level has to be considered as optional. It is best to describe for gaseous streams. If the inorganic-solid global (Dust) is referenced in the gas stream, the dust content has to be defined. If the organic global Char is referenced in the gas stream, the char content has to be defined as well. The same procedure is valid for the organic Tar global. If none of the three globals is referenced, none content has to be defined.

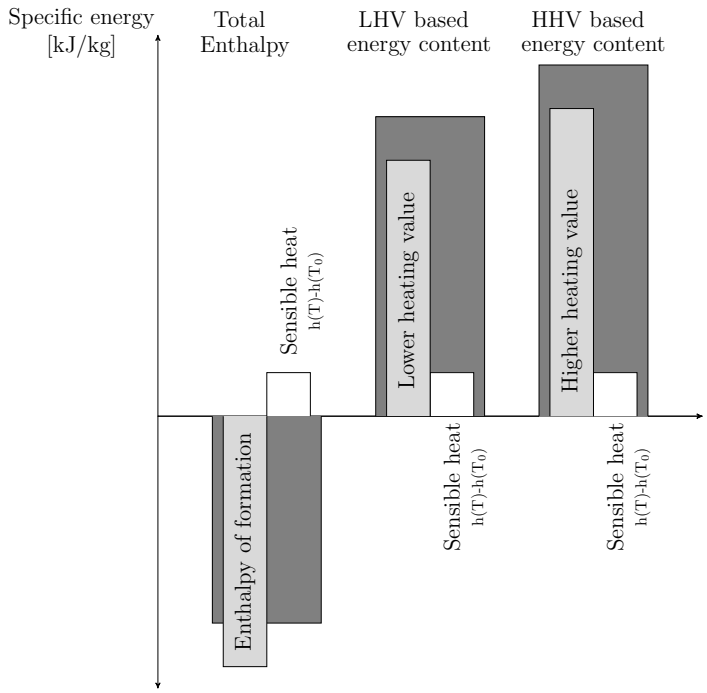


Figure 4.3: Variant diversity of energy balancing [102]

Figure 4.3 shows the energy content of a gasifier product gas stream in three different ways of balancing. All three possibilities have in common the sensible heat term. It is equal for all of them, for its calculation -for gaseous streams- e.g. see Code 4.1 and 4.8.
The very left balance (Fig. 4.3) is built on the total enthalpy, see Equation 4.1 or alternatively Code 4.7 for gas streams.

$$h_{total} = \Delta h^0_{f,298} + h_{p,T} - h_{p_0,T_0} \tag{4.1}$$

$\Delta h^0_{f,298}$ describes the enthalpy of formation at standard conditions (p_0=1 bar, T_0=25 °C), $h_{p,T}$ the thermal enthalpy and h_{p_0,T_0} the thermal enthalpy at standard conditions.[9] [from page 341 and subsequent pages]
For Figure 4.3 the nomenclature is different as it is already considered that for ideal conditions the enthalpy dependency on pressure can be neglected.
All energy balances in the source codes of the unit operations are defined by total enthalpies. Through the implementation of the total enthalpy -respectively the enthalpy of formation- it is not necessary to investigate the reaction enthalpies of the occurring reactions.[104]
However, plenty of elements have a higher level of energy before they are formed to substances, e.g. H and O, when water is formed. Therefore, the reaction is exothermic and its enthalpy of formation is negative. So, negative values for the total enthalpy can be obtained, due to this.
Thus, for an representative energy balance the lower or higher heating value is considered for balancing. In contrary to the total enthalpy, also called conventional or absolute enthalphy, the energy value for gaseous streams is mostly positive for Q_298_and_P.

Nevertheless, negative energy values are possible as well for the Q_298_and_P consideration, most commonly for streams with an high content of water and/or organic compounds. This is because the sensible heat of gaseous connections is based on steam while water and organic compounds are based on liquid water.[102][3.7.2] Therefore, the enthalpy of evaporation is not considered for gas streams (LHV= 0 kJ/kg). This assumption is reasonable as water (steam) is already vaporized in gas streams. In contrary, for pure water- and organic-streams, water is reducing the LHV by its enthalpy of evaporation (LHV= -2442.6 kJ/kg).[102][3.6/3.7/3.8]
Important for all three ways of balancing, also the sensible heat (Q_298) share can cause a negative energy content as it is defined to 298 K (25°C). For temperatures below 25°C the sensible heat term is therefore negative, e.g. for gaseous streams see Code 4.7.

Globals are not visually transparent in the PSE-flowsheet. Globals can be referenced by both other model classes, connections and units. As presented in Figure 4.4. The most common application for globals is defining the chemical composition of mass flows. The benefit of applying one global for several mass flows of the same chemical composition, is having a lower number of variables to be calculated.

Units are representing different process components (gasifier, turbine, ..) by the respective icon. Variables of referenced globals and connections can be used straight in its source code equations. Additional information can be found in [76][5.1.1]. The fundamental equations of an unit's source code are mass- and energy-balances, see [76][4.1] and [102][3.5]. Due to its importance for energy balancing, the definition of the absolute -also called total or conventional- enthalpy is quoted again:

$$h_{total} = \Delta h^0_{f,298} + h_{p,T} - h_{p_0,T_0} \tag{4.1}$$

In Equation 4.1 $\Delta h^0_{f,298}$ is the enthalpy of formation and $h_(p,T) - h_(p_0,T_0)$ describes the share of sensible heat. The standard conditions are considered as p_0= 1 bar and T_0= 25 °C. In contrary to the **Connections** (paragraph before), the energy balance of an unit's source code is created by the absolute enthaply. As the LHV and HHV just consider reactions between the elements C, H, O, N and S. For all the other chemical elements the LHV equals zero. So if $CaCO_3$ is split into CaO and CO_2 within the gasifier's riser model, the released energy would be ignored. And the riser's energy balance would not be fulfilled. [102][3.6] Therefore the energy content of feed and drain stream is preferably defined by the absolute enthalpy in an unit's source code.

4.2 BG_Lib documentation

The idea of the following part is to provide detailed information about the source code of each and every connection, global and unit that can be chosen from the BG_Lib-library. Each of the upcoming sections starts with a table where all the items (variables, globals, functions) are described. Items are the basic elements that are used for writing equations. For a detailed description of items and all the information on how to write or edit a Model Development Kit (MDK)-file, see the MDK documentation [94].

To be able to comprehend the various condition operators (ifl, ref(), !ref(),...) and other operators (&&, !=,...) the MDK-documentation [94] -beginning from page 4-3- should be considered as well. After the description of the items, their mathematical definition is shown. Each item is described separately in a single paragraph named Code.
For reading a MDK-Code it is necessary to understand the interaction between the different model classes. Figure 4.4 shows how the various model classes interact. Unit models can refer to any other model class while globals do not refer to any other model class. So when reading the **Connections** section, cross connections to the **Global** section have to be made.

In each subsection the code, for the calculation of the various items, is listed in the order of item appearance in the table. If necessary some additional explanation is added after the Code lines.
In contrast to other simulation tools that are using a sequential modular solver, IPSEpro is an equation orientated program. This means that the equations of the MDK-file are not calculated in the order they are listed, but all equations are considered as one system of equations.

The described model database is saved under the file name **BG_Lib_2015_06**.

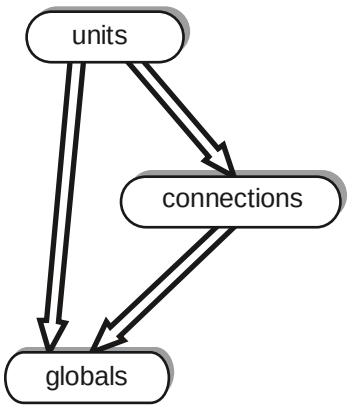


Figure 4.4: Hierarchy of model classes [94]

4.2.1 Connections

Syntax of connections

connection_name.**connection_variable**
e.g.: drain.**massflow**

connection_name.**connection_global**.*global_variable*
e.g.: drain.**Char**.*wC*

connection_name.**connection_global**.*global_function*
e.g.: drain.**Gas**.*gfht*

Each and every connection (stream) has its own name (connection_name). The name of the connection is chosen by the creator of the individual model unit and defined in MDK. In the upcoming sections the variables, globals and functions of the four different connection kinds are described. As already mentioned in this chapter's explanatory preface, there are interactions between the different model classes. In order to make it easier to get them for the global items, in each table of this section the referenced model class global is put in parenthesis, in the item description. The difference between an item-global and a global model class is described in the preface of the **Globals** section.
All function items, either from a connection or global or unit, they are all calling an external Dynamic Link Library (DLL) file. Therefore, there is no item Code, for functions, defined within the IPSEpro model library. Even though their specific meaning is described in the respective tables, it is good to understand their syntax as well and so it is best to already describe it at the beginning of this Subsection.

Syntax of functions

g**f***spt*

The syntax of the exemplary function named **gfspt** provides all the information required. The first letter defines the phase, the function is about. In this example a value for gaseous phase is calculated. Other options are water (w) or organic phase (o). The second letter is always f, it is a short-cut for the word function. The third letter represents the calculated variable of state. The last two letters show the quantities, the variable of state is dependent on. This can either be just one variable or, as shown above, also two. If there are numbers within the function's syntax, they are referring to a certain state is (0 for standard conditions), in this case it is best to take a look at the respective table where the item is described. For Code 4.2 and 4.3 two options are explained, on how to define a function in practice. Based on these two options it should be easy to fulfill any other case.

Shaft

Variable	Description
power	Transferred mechanical energy [kW]

Table 4.2: Shaft connection item description

The shaft connection is applied whenever an electrical device/receiver is connected to a mechanical receiver/device. For example, if an electric motor provides energy to a compressor or if mechanical energy is fed into a generator from a turbine. The model in MDK does not contain any equations, not even for the **power** variable. The power-variable is always calculated within the connected unit models. In case of the electric generator the variable is called shaft_in.power or shaft_out.power, depending on either the mechanical power is an input or output connection.

Gas

Variables	Description
p	Pressure [bar]
t	Temperature [°C]
h	Specific enthalpy of gas stream (sensible heat only) h(0.01°C) = 0.0 [kJ/kg]
s	Specific entropy of gas stream [kJ/kgK]
e	Specific exergy of the gas stream [kJ/kg]
v	Specific volume of gas stream [m^3/kg_gas]
visc	Dynamic viscosity of gas stream, calculation performed only for: CO, CO_2, CH_4, C_2H_4, C_2H_6, C_3H_8, H_2O, H_2, N_2 and O_2 [Pas·10^{-6}]
lambda	Thermal conductivity of gas stream, calculation performed only for: CO, CO_2, CH_4, C_2H_4, C_2H_6, C_3H_8, H_2O, H_2, N_2 and O_2 [W/(mK)]
q_298_total	Gas mass specific sensible heat from whole gas stream based to 25 °C (q_298_total (25°C) = 0.0)
h_total	Total thermochemical enthalpy (enthalpy of formation + sensible heat) of the gas stream (gas mass specific) [kJ/kg_gas]
lhv_total	Lower heating value of the whole gas stream (gas mass specific) [kJ/kg_gas]
hhv_total	Higher heating value of the whole gas stream (gas mass specific) [kJ/kg_gas]
mass flow	Mass flow of the carrier gas stream. Mass flow is the base for all specific quantities [kg_gas/h]
Mass_total	Total mass flow including mass of dust, char, and tar loads on the gas stream. Mass_total is NOT the base for specific quantities [kg/h]
nvolflow	Total volume flow of the gas stream at standard conditions (1.01325 bar, 0.0°C) [Nm^3/h]
opvolflow	Total volume flow of the gas stream at operating conditions [m^3/h]
Energy	Total energy (heat, pressure, and chemical) transported by the stream based on lower heating value (enthalpy of vaporisation of condensable water contained is not considered) [kW]

Variables	Description
Energy_hhv	Total energy (heat, pressure, and chemical) transported by the stream based on higher heating value (enthalpy of vapourisation of condensable water contained is considered) [kW]
Exergy	Total exergy transported by the stream based on thermodynamic equilibrium environment according to Baehr and Diederichsen [kW]
dust_content	Dust content on the gas stream. If global object Dust is not referenced, dust_content is set to 0.0 by the gas connection model [g/Nm^3]
h_dust	Specific sensible enthalpy of dust in kJ/kg_dust. If global object Dust is not referenced, h_dust is set to 1.0 by the gas connection model
s_dust	Specific entropy of dust in kJ/(kg_dust.K). If global object Dust is not referenced, s_dust is set to 1.0 by the gas connection model
e_dust	Specific exergy of dust in kJ/kg_dust. If global object Dust is not referenced, e_dust is set to 1.0 by the gas connection model
char_content	Char content on the gas stream. If global object Char is not referenced, tar_content is set to 0.0 by the gas connection model [g/Nm^3]
h_char	Specific sensible enthalpy of char in kJ/kg_dust. If global object Char is not referenced, h_char is set to 1.0 by the gas connection modell
s_char	Specific entropy of char in kJ/(kg_dust.K). If global object Char is not referenced, s_char is set to 1.0 by the gas connection model
e_char	Specific exergy of char in kJ/kg_dust. If global object Char is not referenced, e_char is set to 1.0 by the gas connection model
tar_content	Tar content on the gas stream. If global object Tar is not referenced, tar_content is set to 0.0 by the gas connection model [g/Nm^3]
h_tar	Specific sensible enthalpy of gaseous tar in kJ/kg_tar. If global object Tar is not referenced, h_tar is set to 1.0 by the gas connection model
s_tar	Specific entropy of gaseous tar in kJ/(kg_tar.K). If global object Tar is not referenced, s_tar is set to 1.0 by the gas connection model
e_tar	Specific exergy of gaseous tar in kJ/kg_tar. If global object Tar is not referenced, e_tar is set to 1.0 by the gas connection model
Functions	**Description**
wfp0t	Returns the vapour pressure of water at a certain temperature from simprop library [bar]
Globals	**Description**
Gas	Chemical composition of the carrier gas stream (referencing to gas global)
Dust	Optional global object defining the composition of an inorganic dust load on the stream (referencing to solid global)
Char	Optional global object defining the composition of a char coke load on the stream (referencing to organic global)
Tar	Optional global object defining the composition of a tar load on the stream (referencing to organic global)

Table 4.3: Gas connection item description

The variables *p*, *t* and *massflow* are connected to the unit at the beginning or the end of the connection and therefore either calculated by them or simply set but not calculated by an equation within the connections source code. For the formulaic description of the other variables see the following code equations. They are listed in order of appearance in Table 4.3.

Code 4.1: Specific enthalpy of gas stream

```
fh:      h = Gas.gfht(t);
```

Code 4.2: Specific entropy of gas stream

```
fs:      s = Gas.gfspt(p, t);
```

The *gfspt* function of Code 4.2 is calculated in dependence of the current pressure and temperature while in Code 4.3 it is calculated for a fixed pressure of 1 bar and a fixed temperature of 25°C.

Code 4.3: Specific exergy of gas stream

```
fe:      e = Gas.gfe0() + h - Gas.gfht(25.0) - 298.15 * (s - Gas.
    gfspt(1.0, 25.0));
```

Code 4.4: Specific volume of the carrier gas stream

```
fv:      v * Gas.M * p * 100.0 = 8.31451 * (t + 273.15);
```

Code 4.5: Dynamic viscosity

```
fvisc: visc = Gas.gfvt(t)*1000000;
```

Code 4.6: Thermal conductivity λ

```
flambda: lambda = Gas.gflt(t);
```

Additional theoretical information on the dynamic viscosity and thermal conductivity is provided by J. Kotik [60][3.2.1.1 to 3.2.1.5].

Code 4.7: Only Gas, neither Dust nor Char nor Tar

```
ifl !ref(Dust) && !ref(Char) && !ref(Tar) then
fh_total1: h_total = h + Gas.gfhf273();
f_q_298_total1: q_298_total = (h - Gas.gfht(25));
f_lhv_total1: lhv_total = Gas.LHV * 10^6 * (8.31451*273.15/(Gas.M
    *101325.0));
f_hhv_total1: hhv_total = Gas.HHV * 10^6 * (8.31451*273.15/(Gas.M
    *101325.0));
endifl
```

Code 4.7 shows the calculation of the variables *q_298_total*, *h_total*, *lhv_total* and *hhv_total* for the special case of just gas, neither dust nor char nor tar is referenced for the gas connection. In total 8 different cases are considered for including or excluding dust, char or tar. For reasons of clarity they are not listed in this section but can be seen in the BG_Lib MDK-file.

Code 4.8: Total mass flow of gas including dust, char and tar loads

```
fMass_total:    Mass_total = massflow + nvolflow*0.001*(
    dust_content + char_content + tar_content);
```

Code 4.9: Normal volume flow at normal conditions: t = 273.15 K, p = 1.01325 bar

```
fnvolflow:      nvolflow * Gas.M * 101.325 = massflow * 8.31451 *
    273.15;
```

Code 4.10: Operating volume flow opvolflow

```
fopvolflow:     opvolflow = massflow * v;
```

Code 4.11: Energy based on specific sensible heat and lower heating value

```
fEnergy:        Energy * 3600 = massflow * (lhv_total +
    q_298_total);
```

Code 4.12: Energy based on specific sensible heat and higher heating value

```
fEnergy_hhv:    Energy_hhv * 3600 = massflow * (hhv_total +
    q_298_total);
```

Code 4.13: Exergy transported by the entire stream

```
fExergy:        Exergy*3600.0    = massflow * e
                + nvolflow*dust_content*1.0e-3*e_dust
                + nvolflow*char_content*1.0e-3*e_char
                + nvolflow*tar_content*1.0e-3*e_tar;
```

Code 4.14: Dust equations section

```
ifl ref(Dust) then
        fh_dust:        h_dust = Dust.sfht(t);
        fs_dust:        s_dust = Dust.sfst(t);
        fe_dust:        e_dust = Dust.e_chem + h_dust - Dust.sfht
            (25.0) - 298.15 * (s_dust - Dust.sfst(25.0));
elsel
        fno_dust:       dust_content = 0.0;
        fh_no_dust:     h_dust = 1.0;
        fs_no_dust:     s_dust = 1.0;
        fe_no_dust:     e_dust = 1.0;
endifl
```

Code 4.15: Char equations section

```
ifl ref(Char) then
        fh_char:        h_char = Char.ofht(t);
        fs_char:        s_char = Char.ofst(t);
        fe_char:        e_char = Char.e_chem + h_char - Char.ofht
            (25.0) - 298.15 * (s_char - Char.ofst(25.0));
elsel
```

```
        fno_char:       char_content = 0.0;
        fh_no_char:     h_char = 1.0;
        fs_no_char:     s_char = 1.0;
        fe_no_char:     e_char = 1.0;
endifl
```

Code 4.16: Tar equations section

```
ifl ref(Tar) then
        fh_tar: h_tar = Tar.ofht(t);
        fs_tar: s_tar = Tar.ofst(t);
        fe_tar: e_tar = Tar.e_chem + h_tar - Tar.ofht(25.0) -
            298.15 * (s_tar - Tar.ofst(25.0));
elsel
        fno_tar:        tar_content = 0.0;
        fh_no_tar:      h_tar = 1.0;
        fs_no_tar:      s_tar = 1.0;
        fe_no_tar:      e_tar = 1.0;
endifl
```

For Code 4.14, 4.15 and 4.16, if no dust/char/tar are present all the variable values are set to 1, not to bother the solution process.

Organic

Variables	Description
p	Pressure [bar]
t	Temperature [°C]
h_waf	Specific sensible enthalpy of organic substance free of water and ash [kJ/kg]
s_waf	Specific entropy of organic substance free of water and ash [kJ/kgK]
e_waf	Specific exergy of organic substance free of water and ash [kJ/kg]
water_content	Water content in total mixture [kg/kg_total]
ash_content	Ash content in total mixture [kg/kg_total]
q_298_total	Mass specific sensible heat from whole stream based to 25 °C (q_298_total (25°C) = 0.0)
h_total	Total thermodynamic enthalpy of the entire stream based on total mass units (organic + water + ash) [kJ/kg_total]
lhv_total	Lower heating value of the entire mixture (organic + water + ash) [kJ/kg_total]
hhv_total	Higher heating value of the entire mixture (organic + water + ash) [kJ/kg_total]
massflow	Total mass flow (organic + water + ash) [kg_total/h]
volflow	Total volume flow (organic + water + ash) [m^3/h]
Energy	Energy transported by the entire stream (sensible enthalpy based to 25 °C + lower heating value) [kW]
Energy_hhv	Energy transported by the entire stream (sensible enthalpy based to 25 °C + higher heating value) [kW]
Exergy	Exergy transported by the entire stream [kW]
Functions	**Description**
wftph	Temperature as a function of pressure and enthalpy [°C]
wfsph	Specific entropy as a function of pressure and enthalpy [kJ/kgK]
wfvph	Specific volume as a function of pressure and enthalpy [m^3/kg]
wfhpt	Specific sensible enthalpy as a function of pressure and temperature [kJ/kg]
Globals	**Description**
Organic	Elementary composition of the organic substance (referencing to organic global)
Ash	Composition of inorganic substances added to the organic stream (referencing to solid global)

Table 4.4: Organic connection item description

Code 4.17: Specific sensible enthalpy of the organic substance free of water and ash

```
fh:     h_waf = Organic.ofht(t);
```

Code 4.18: Specific entropy of the organic substance free of water and ash

```
fs:     s_waf = Organic.ofst(t);
```

Code 4.19: Specific exergy of the organic substance free of water and ash

```
fe:     e_waf = Organic.e_chem + h_waf - Organic.ofht(25.0) -
    298.15*(s_waf - Organic.ofst(25.0));
```

Code 4.20: Specific total enthalpy and sensible heat in case of referenced ash

```
# Case 1: No Ash referenced
ifl !ref(Ash) then
                                    .
                                    .
# Case 2: Ash referenced
elsel
fh_total2:      if water_content <= 0.000001 then
        h_total = (1.0 - ash_content)*(h_waf + Organic.hf298 -
            Organic.ofht(25.0) + Organic.ofht(0.0)) + ash_content*(
            Ash.sfht(t) + Ash.sfhf273());
else
        h_total = (1.0 - water_content - ash_content)*(h_waf +
            Organic.hf298 - Organic.ofht(25.0) + Organic.ofht(0.0))
             + water_content*((-285830.0/18.01528) - wfhpt(1.00000,
             25.0) + wfhpt(p,t))+ ash_content*(Ash.sfht(t) + Ash.
            sfhf273());

fq_298_total2:  if water_content <= 0.000001 then
        q_298_total = (1.0 - ash_content) * (h_waf - Organic.ofht
            (25.0) + Organic.ofht(0.0)) + ash_content * (Ash.sfht(t
            ) - Ash.sfht(25.0));
else
        q_298_total = (1.0 - water_content - ash_content) * (h_waf
             - Organic.ofht(25.0) + Organic.ofht(0.0)) +
            water_content * (wfhpt(p,t) - wfhpt(1.00000,25.0)) +
            ash_content * (Ash.sfht(t) - Ash.sfht(25.0));
endifl
```

If no ash is referenced (Case 1) the ash containing terms are missing but the rest remains equal and is therefore not presented.

Code 4.21: Lower heating value of the total mixture (organic + water + ash)

```
flhv:    lhv = (1.0 - water_content - ash_content)*Organic.lhv -
    water_content*2442.6;
```

Code 4.22: Higher heating value of the total mixture (organic + water + ash)

```
fhhv:    hhv = (1.0 - water_content - ash_content)*Organic.hhv;
```

Code 4.23: Volume flow

```
fvolflow:        rho*volflow = massflow;
```

Code 4.24: Energy transported by the entire stream

```
fEnergy:         Energy*3600.0 = massflow*(q_298_total + lhv);
fEnergy_hhv:     Energy_hhv*3600.0 = massflow*(q_298_total + hhv);
```

Code 4.25: Exergy transported when ash is referenced

```
# Case 1: No Ash referenced
ifl !ref(Ash) then
                                    .
                                    .
# Case 2: Ash referenced
elsel
fExergy2:        if water_content <= 0.000001 then
        Exergy*3600.0    = massflow*((1.0 - ash_content)*(h_waf -
            Organic.ofht(25.0) - 298.15*(s_waf - Organic.ofst(25.0)
            ) + Organic.e_chem) + ash_content*(Ash.e_chem + Ash.
            sfht(t) - Ash.sfht(25.0) - 298.15*(Ash.sfst(t) - Ash.
            sfst(25.0))));
else
        Exergy*3600.0    = massflow*((1.0 - water_content -
            ash_content)*(h_waf - Organic.ofht(25.0) - 298.15*(
            s_waf - Organic.ofst(25.0)) + Organic.e_chem) +
            water_content*(wfhpt(p,t) - 104.85 - 298.15 * (wfsph(p,
            wfhpt(p,t)) - 3.8822044) + 1.62) + ash_content*(Ash.
            e_chem + Ash.sfht(t) - Ash.sfht(25.0) - 298.15*(Ash.
            sfst(t) - Ash.sfst(25.0))));
endifl
ifl !ref(Ash) then fash_cont: ash_content = 0.0;
endifl
```

Solid

The solid connection class is also declared as inorganic stream class.

Variables	Description
t	Temperature of the solid stream [°C]
h	Specific sensible enthalpy of the inorganic solid material [kJ/kg]
s	Specific entropy of the inorganic solid material [kJ/kgK]
e	Specific exergy of the inorganic solid material [kJ/kg]
q_298_total	Specific sensible enthalpy of the whole stream, q_298_total = 0 at 25 °C
h_total	Total specific enthalpy of the entire stream including enthalpy of formation and total enthalpies of char and gas. The quantity is specific to mass units of inorganic solid material [kJ/kg_inorg]
lhv_total	Lower heating value of the whole stream [kJ/kg_inorg]
hhv_total	Higher heating value of the whole stream [kJ/kg_inorg]
massflow	Mass flow of inorganic solid substance [kg_inorg/h]
Mass_total	Total mass flow of solid stream including mass of organic char and adsorbed gas loads. Mass_total is NOT the base for specific quantities and not used for mass balancing! [kg_total/h]
Energy	Total energy (heat, pressure, and chemical) transported by the stream based on lower heating value (enthalpy of vapourisation of condensable water contained is not considered) [kW]
Energy_hhv	Total energy (heat, pressure, and chemical) transported by the stream based on higher heating value (enthalpy of vapourisation of condensable water contained is considered) [kW]
Exergy	Total exergy transported by the solid stream [kW]
char_content	Char load on the inorganic stream. If no global object Char is selected, char_content is set to 0.0 by the connection solid model [kg/kg_inorg]
h_char	Specific sensible enthalpy of char [kJ/kg]
s_char	Specific entropy of char [kJ/kgK]
e_char	Specific exergy of char [kJ/kg]
gas_ads	Gas load on the inorganic solid stream [mol/kg_inorg]
H_gas	Molar enthalpy of incorporated gas [kJ/mol]
S_gas	Molar entropy of incorporated gas [J/molK]
E_gas	Molar exergy of incorporated gas [kJ/mol]
Globals	**Description**
Solid	Composition of the inorganic solid mixture (referencing to solid global)
Char	Optional global defining the composition of organic substances added to the solid stream (referencing to organic global)
Gas	Optional global defining the composition of gas incorporated in the solid stream (referencing to gas global)

Table 4.5: Solid connection item description

Code 4.26: Specific sensible enthalpy of the inorganic solid material

```
fh:      h = Solid.sfht(t);
```

Code 4.27: Specific entropy of the inorganic solid material

```
fs:      s = Solid.sfst(t);
```

Code 4.28: Specific exergy of the inorganic solid material

```
fe:      e = Solid.e_chem + h - Solid.sfht(25.0) - 298.15*(s -
    Solid.sfst(25.0));
```

Code 4.29: Only inorganic solid, neither Char nor Gas

```
         ifl !ref(Char) && !ref(Gas) then
fh_total1:       h_total = h + Solid.sfhf273();
fq_298_total1:   q_298_total = (h - Solid.sfht(25));
flhv_total1:     lhv_total = 0;
fhhv_total1:     hhv_total = 0;
         endifl
```

Code 4.29 shows the case of no char and gas in the solid stream. In total four different cases of solid-gas-tar combinations are considered.

Code 4.30: Total mass flow of solid stream including organic char and adsorbed gas loads

```
         ifl !ref(Gas) then
f1Mass_total:    Mass_total = massflow*(1.0 + char_content);
         elsel
f2Mass_total:    Mass_total = massflow*(1.0 + char_content +
    gas_ads*0.001*Gas.M);
         endifl
```

The first option of Code 4.30 assumes that no gas is adsorbed in the inorganic solid stream, for the second option there is gas adsorbed.

Code 4.31: Energy transported by the entire stream

```
fEnergy:         Energy * 3600.0 = massflow * (q_298_total +
    lhv_total);
fEnergy_hhv:Energy_hhv * 3600.0 = massflow * (q_298_total +
    hhv_total);
```

Code 4.32: Exergy transported by the entire solid stream

```
         ifl ref(Gas) then
fExergy1:        Exergy*3600.0    = massflow*(e + char_content*
    e_char + gas_ads*E_gas);
         elsel
fExergy2:        Exergy*3600.0    = massflow*(e + char_content*
    e_char);
         endifl
```

Code 4.33: Char equation section

```
        ifl ref(Char) then
fh_char: h_char = Char.ofht(t);
fs_char: s_char = Char.ofst(t);
fe_char: e_char = Char.e_chem + Char.ofht(t) - Char.ofht(25.0) -
    298.15*(Char.ofst(t) - Char.ofst(25.0));
        elsel
fchar_cont_no_char: char_content = 0.0;
fh_no_char: h_char = 1.0;
fs_no_char: s_char = 1.0;
fe_no_char: e_char = 1.0;
        endifl
```

The values of 1.0 -for the second case of Code 4.33- are just defined to not bother the solution process, for the case that no char is referenced.

Code 4.34: Gas equation section

```
        ifl ref(Gas) then
fH_gas: H_gas = 1.0e-3*Gas.M*Gas.gfht(t);
fS_gas: S_gas = Gas.M*Gas.gfspt(1.0, t);
fE_gas: E_gas = 1.0e-3*Gas.M*(Gas.gfe0() + Gas.gfht(t) - Gas.gfht
    (25.0) - 298.15*(Gas.gfspt(1.0, t) - Gas.gfspt(0.91771, 25.0)))
    ;
        elsel
fgas_ads_no_gas: gas_ads = 0.0;
fH_no_gas: H_gas = 1.0;
fS_no_gas: S_gas = 1.0;
fE_no_gas: E_gas = 1.0;
        endifl
```

The molar exergy of the adsorbed gas is defined according to [9] p. 355, the equilibrium conditions are chosen as 25.0 °C and 0.91771 bar. No pressure is defined for solid streams as it is neither relevant for the adsorbed gases, the pressure for gas property calculation is simply set to 1.0 bar, within Code 4.34.

Water

Variables	Description
p	Pressure of the water stream [bar]
t	Temperature of the water stream [°C]
h	Specific enthalpy of working fluid stream (sensible heat only) h = 0.0 at triple point of the medium (Water: 0.01°C, 0.00611 bar) [kJ/kg]
s	Specific entropy of the working fluid stream [kJ/kgK]
e	Specific exergy of the working fluid stream [kJ/kg]
v	Specific volume of the working fluid stream [m^3/kg]
x_steam	Steam quality (x = 0 for pure liquid, x = 1 for superheated steam) of the working fluid [kg/kg]
q_298_total	Specific enthalpy of working fluid stream (sensible heat only). q_298_total = 0.0 at 25°C and 1.00000 bar [kJ/kg]
h_total	Total thermochemical enthalpy (enthalpy of formation + sensible heat) of the working fluid stream (mass specific) [kJ/kg]
massflow	Mass flow of working fluid [kg/h]
Energy	Total energy (heat, pressure, and chemical) transported by the stream based on lower heating value (enthalpy of vapourisation of condensable water contained is not considered) [kW]
Energy_hhv	Total Energy (heat, pressure, and chemical) transported by the stream based on higher heating value (enthalpy of vapourisation of condensable water contained is considered) [kW]
Exergy	Total exergy transported by the stream based on thermodynamic equilibrium environment according [9] p.355 [kW]
Functions	**Description**
wftph	Temperature as a function of pressure and enthalpy
wfsph	Specific entropy as a function of pressure and enthalpy
wfvph	Specific volume as a function of pressure and enthalpy
wfxph	Steam quality (steam fraction) as a function of pressure and enthalpy
wfhpx	Specific sensible enthalpy as a function of pressure and steam quality (steam fraction)
wfhpt	Specific sensible enthalpy as a function of pressure and temperature
wfp0t	Returns vapour pressure of water at a certain temperature

Table 4.6: Water connection item description

Code 4.35: Specific sensible enthalpy/temperature of the working fluid stream

```
ft:     if blocksize() == 1.0 && isconverged(p) && isconverged(t)
    then
                h = wfhpt(p, t);
        else
                t = wftph(p, h);
```

In Code 4.35 it is written, if the number of variables/equations -returned by the blocksize function- equals 1 and the pressure- and temperature-values converge, the enthalpy is cal-

culated by the function *wfhpt* in dependence of temperature and pressure. Otherwise the temperature is calculated in dependence of pressure and enthalpy.

Code 4.36: Specific entropy s of the working fluid stream

```
fs:     s = wfsph(p, h);
```

Code 4.37: Specific exergy of the working fluid stream

```
fe:     e =  1.62 + (h - 104.928) - 298.15 * (s - 3.8822044);
```

In Code 4.37 the value 1.62 describes the chemical exergy of water at standard conditions (1 bar, 25°C), according to property data calculations of the creator of this unit, M. Stidl.

Code 4.38: Specific volume v of the working fluid

```
fv:     v = wfvph(p,h);
```

Code 4.39: Steam quality x_steam of the working fluid

```
fx_steam:       if (isconverged(p) && isconverged(h))
        then
                x_steam = wfxph(p, h);
        else
                h - wfhpx(p, 0.0) - x_steam*(wfhpx(p, 1.0) -
                    wfhpx(p, 0.0)) = 0.0;
```

Code 4.40: Total thermodynamic enthalpy h_total of the working fluid

```
fh_total: h_total = (-285830.0/18.01528) - wfhpt(1.00000, 25.0) +
    h;
```

In Code 4.40 *h_total* includes the enthalpy of formation which is 0 for the thermodynamically stable modification of a pure element at 25°C (298.15 K). The enthalpy of formation of water is converted from 285830 J/mol (at 298.15 K, 1.0 bar) into kJ/kg. The sensible enthalpy difference is calculated from the actual thermal conditions *h* (Code 4.35) to standard conditions. *h_total* should be used in overall heat balances for chemical reactors.

Code 4.41: Specific sensible heat at 25 °C (298,15 K) and 1.0 bar of the whole stream

```
fq_298_total:   q_298_total = (h - wfhpt(1.00000,25.0));
```

Code 4.42: Energy (based on h and lhv or higher hhv) transported by the water stream

```
fEnergy:        Energy*3600.0 = massflow * (q_298_total - 2442.6);
fEnergy_hhv:    Energy_hhv*3600.0 = massflow * (q_298_total);
```

Code 4.43: Exergy transported by the entire stream

```
fExergy:        Exergy*3600.0 = massflow * e;
```

4.2.2 Globals

In this section the source code of the four different global model classes is described. When creating a connection or unit model, there is always the option of referring to one of this four global model classes by adding an item which is also called global. In this library they are named e.g. *Tar*, *Ash*, *Gas*, *AmbAir* and so on. All of this item-globals are referring to one of the following global model source codes.

Ambient

Variables	Description
t	Ambient temperature used for ambient air. This ambient temperature does not affect the values for exergy, where always 25°C are used as the reference temperature [°C]
p	Ambient air pressure correlated to the geodetic altitude [bar]
h	Altitude above sea level [m]
phi_rel	Relative humidity of ambient air at ambient conditions [%]

Table 4.7: Ambient global item description

Code 4.44: Pressure - height relation

```
f_p:      p = 1.01325*(1 - 0.0065*h/288.15)^5.2558;
```

Code 4.44 shows the relationship between altitude and pressure according to ISO 2533. The other variables of Table 4.7 have to be defined manually.

Gas

Variables	Description
wAR	Mass fraction of Ar (Argon)
wC2H4	Mass fraction of C_2H_4 (Ethylene)
wC2H6	Mass fraction of C_2H_6 (Ethane)
wC3H8	Mass fraction of C_3H_8 (Propane)
wCH4	Mass fraction of CH_4 (Methane)
wCO	Mass fraction of CO (Carbon oxide)
wCO2	Mass fraction of CO_2 (Carbon dioxide)
wH2	Mass fraction of H_2 (Hydrogen)
wH2O	Mass fraction of H_2O(g) (Water)
wH2S	Mass fraction of H_2S (Hydrogen sulfide)
wHCl	Mass fraction of HCl (Hydrogen chloride)
wHCN	Mass fraction of HCN (Hydrogen cyanide)
wN2	Mass fraction of N_2 (Nitrogen)
wN2O	Mass fraction of N_2O (Nitrous oxide)
wNH3	Mass fraction of NH_3 (Ammonia)

Variables	Description
wNO	Mass fraction of NO (Nitrogen oxide)
wO2	Mass fraction of O_2 (Oxygen)
wSO2	Mass fraction of SO_2 (Sulfur dioxide)
yAR	Mole fraction of Ar (Argon)
yC2H4	Mole fraction of C_2H_4 (Ethylene)
yC2H6	Mole fraction of C_2H_6 (Ethane)
yC3H8	Mole fraction of C_3H_8 (Propane)
yCH4	Mole fraction of CH_4 (Methane)
yCO	Mole fraction of CO (Carbon oxide)
yCO2	Mole fraction of CO_2 (Carbon dioxide)
yH2	Mole fraction of H_2 (Hydrogen)
yH2O	Mole fraction of H_2O(g) (Water)
vH2S	Mole fraction of H_2S (Hydrogen sulfide)
yHCl	Mole fraction of HCl (Hydrogen chloride)
yHCN	Mole fraction of HCN (Hydrogen cyanide)
yN2	Mole fraction of N_2 (Nitrogen)
yN2O	Mole fraction of N_2O (Nitrous oxide)
yNH3	Mole fraction of NH_3 (Ammonia)
vNO	Mole fraction of NO (Nitrogen oxide)
yO2	Mole fraction of O_2 (Oxygen)
ySO2	Mole fraction of SO_2 (Sulfur dioxide)
M	Molar weight of the gas mixture [kg/kmol]
Hf298	Standard enthalpy of formation at 298.15K. For the thermodynamic stable modification of a pure element at 298.15K Hf298 = 0 [kJ/mol]
LHV	Lower heating value (net calorific value) of the gas mixture [MJ/Nm^3]
HHV	Higher heating value (gross calorific value) of the gas mixture [MJ/Nm^3]
Function	**Description**
gfht	Function returns the specific enthalpy (sensible heat only, h(0.01°C) = 0.0) of a gas mixture defined by the gas composition
gfvt	Function returns the dynamic viscosity in [Pas] for specific gas components of a gas stream (CO, CO_2, CH_4, C_2H_4, C_2H_6, C_3H_8, H_2O, H_2, N_2 and O_2)
gflt	Function returns the thermal conductivity in [W/(mK)] for specific gas components of a gas stream (CO, CO_2, CH_4, C_2H_4, C_2H_6, C_3H_8, H_2O, H_2, N_2 and O_2)
gfspt	Function returns the specific entropy of the gas mixture defined by the gas composition
gfhf273	Function returns the specific standard enthalpy of formation at 273.15K of the gas mixture defined by the gas composition
gfHf298	Function returns the molar standard enthalpy of formation at 298.15K of the gas mixture defined by the gas composition
gfe0	Function provides the chemical exergy of the ideal gas mixture considering a thermodydnamic equilibrium environment according to [9]

Table 4.8: Gas global item description

Code 4.45: Sum of molar fractions

```
fmolsum: yAr + yC2H4 + yC2H6 + yC3H8 + yCH4 + yCO + yCO2 + yH2 +
    yH2O + yH2S + yHCl + yHCN + yN2 + yN2O + yNH3 + yNO + yO2 +
    ySO2 = 1.0;
```

Code 4.46: Molar weight of gas mixture

```
fM:      M       = yAr * 39.9480 + yC2H4 * 28.0536 + yC2H6 *
    30.0694 + yC3H8 * 44.0962 + yCH4 * 16.0428 + yCO * 28.0104 +
    yCO2 * 44.0098 + yH2 * 2.01588 + yH2O * 18.01528 + yH2S *
    34.08188 + yHCl * 36.46064 + yHCN * 27.02568 + yN2 * 28.01348 +
     yN2O * 44.01288 + yNH3 * 17.03056 + yNO * 30.00614 + yO2 *
    31.9988 + ySO2 * 64.0648;
```

The molar weight of the single species (Code 4.46/4.47) is according to [7].

Code 4.47: Mass fraction / mole fraction conversion

```
fAr:    wAr  *   M =    yAr  *   39.9480;
fC2H4:  wC2H4 *  M =    yC2H4 *  28.0536;
fC2H6:  wC2H6 *  M =    yC2H6 *  30.0694;
fC3H8:  wC3H8 *  M =    yC3H8 *  44.0962;
fCH4:   wCH4 *   M =    yCH4 *   16.0428;
fCO:    wCO  *   M =    yCO  *   28.0104;
fCO2:   wCO2 *   M =    yCO2 *   44.0098;
fH2:    wH2  *   M =    yH2  *   2.01588;
fH2O:   wH2O *   M =    yH2O *   18.01528;
fH2S:   wH2S *   M =    yH2S *   34.08188;
fHCl:   wHCl *   M =    yHCl *   36.46064;
fHCN:   wHCN *   M =    yHCN *   27.02568;
fN2:    wN2  *   M =    yN2  *   28.01348;
fN2O:   wN2O *   M =    yN2O *   44.01288;
fNH3:   wNH3 *   M =    yNH3 *   17.03056;
fNO:    wNO  *   M =    yNO  *   30.00614;
fO2:    wO2  *   M =    yO2  *   31.9988;
fSO2:   wSO2 *   M =    ySO2 *   64.0648;
```

Code 4.48: Molar standard enthalpy of formation

```
fHf298: Hf298 = gfHf298() * 1.0e-3;
```

The molar standard enthalpy of formation is returned from the SimProp.dll-file in J/mol and therefore converted via the equation of Code 4.48 to gain the unit kJ/mol.

Code 4.49: Lower heating value LHV

```
fLHV:    LHV = yC2H4 * 59.0238 + yC2H6 * 63.6936 + yC3H8 * 91.1917
    +  yCH4 * 35.7932 +  yCO * 12.6251 + yH2 * 10.7890 +  yH2S *
    23.1132 + yHCN * 28.9739 + yN2O * 3.6606 + yNH3 * 14.1339 + yNO
     * 4.0719;
```

The lower heating value -energy released by complete combustion of a fuel unit with free oxygen [104]- is defined in MJ/Nm^3 according to DIN 51857. The coefficients are generated through the enthalpy of formation, taken from [7]. For gaseous streams the enthalpy

of formation is in general applied to calculate the reaction enthalpy of the combustion. The reaction enthalpy is equal to the negative value of the molar lower heating value (LHV). Based on this, the *LHV* -on standard volume base- is obtained. A detailed explanation on this calculation is given by [9] p.469, Example 7.4.

Code 4.50: Higher heating value HHV

```
fHHV:    HHV = yC2H4 * 62.9503 + yC2H6 * 69.5833 + yC3H8 * 99.0446
    +  yCH4 * 39.7196 +  yCO * 12.6251 + yH2 * 12.7522 +  yH2S *
    25.0764 + yHCN * 29.9555 + yN2O * 3.6606 + yNH3 * 17.0788 + yNO
     * 4.0719 + yH2O * 1.9632;
```

In comparison to Code 4.49, in Code 4.50 the enthalpy of formation for liquid H_2O is considered. Therefore, the higher heating value (*HHV*) is calculated. This results in different coefficients for species which are based on the element hydrogen. Also this procedure is described by [9] p.469, Example 7.4. The calculation of *LHV* and *HHV* is further described in [104] p.22/23.

Organic

Switch	Description
ModelSubstance	Switch determines the applied model substance for thermodynamic data such as h, s, and e_chem.
hvSource	Switch defines whether the lhv should set to a default value for a mixture of basic tar components at 298.15 K or if it will be provided by the user
hf298Method	Switch defines whether the specific enthalpy of formation should be calculated from lhv or set to default value for a mixture of basic tar components
Variables	**Description**
SubstNr	Code for the DLL defining the applied model substance. 1 = char 2 = wood 10 = tar 20 = light fuel oil 21 = rape oil methyl ester (RME) 22 = heat carrier oil (MobilTherm)
wC	Mass fraction of carbon in organic composition
wH	Mass fraction of hydrogen in organic composition
wO	Mass fraction of oxygen in organic composition
wN	Mass fraction of nitrogen in organic composition
wS	Mass fraction of sulfur in organic composition
wCl	Mass fraction of chlorine in organic composition
lhv	Lower heating value of organic mixture at 298.15 K (solid tar) [kJ/kg]
hhv	Higher heating value of organic mixture at 298.15 K (solid tar) [kJ/kg]
hf298	Standard specific enthalpy of formation of the gaseous-tar mixture at 298.15K. For the thermodynamic stable modification of a pure element at 298.15K hf298 = 0 [kJ/kg]
e_chem	Specific exergy of the tar mixture at environmental conditions (25°C, 1 bar) [kJ/kg]
Functions	**Description**
ofht	Function provides the specific sensible enthalpy of the organic mixture
ofst	Function provides the specific entropy of the default mix in SimProp

Table 4.9: Organic global item description

Code 4.51: Setting the applied model substance for the thermodynamic data imported from SimProp DLL

```
ifl ModelSubstance == Char then fSubstNr1: SubstNr = 1.0; endifl
ifl ModelSubstance == Wood then fSubstNr2: SubstNr = 2.0; endifl
ifl ModelSubstance == Graphite then fSubstNr3: SubstNr = 3.0;
    endifl
ifl ModelSubstance == LightFuelOil then fSubstNr20: SubstNr =
    20.0; endifl
ifl ModelSubstance == RME then fSubstNr21: SubstNr = 21.0; endifl
ifl ModelSubstance == MobilTherm then fSubstNr22: SubstNr = 22.0;
    endifl
ifl ModelSubstance == Therminol66 then fSubstNr23: SubstNr = 23.0;
     endifl
```

```
ifl ModelSubstance == OLGA then fSubstNr25: SubstNr = 25.0; endifl
ifl ModelSubstance == Tar then fSubstNr10: SubstNr = 10.0; endifl
```

According to the selected substance of Code 4.51, the auxiliary variable *SubstNr* will be set and implicitly integrated in the call functions for the SimProp DLL file.

Code 4.52: Sum of mass fractions

```
fmass_sum:      wC + wH + wO + wN + wS + wCl = 1.0;
```

Code 4.53: Lower heating value

```
ifl hvSource == Boie then
        flhv_boie:      lhv = wC*34835.0 + wH*93870.0 - wO*10800.0
            + wN*6280.0 + wS*10465.0;
endifl
ifl hvSource == TarDefaultMix then
        flhv_tar:       lhv = 38489.5;
endifl
```

The boie correlation is based on an ultimate substance analysis. It is applied as most organic fuels (biomass, coal,...) are complex mixtures of an undefined molecular composition, in comparison to common gaseous fuels (e.g. methane) [104] p.23.

Code 4.54: Higher heating value

```
fhhv:   hhv = lhv + wH*21828.68;
```

The higher heating value (HHV) is calculated as a function of *lhv* and hydrogen, based on enthalpies of formation [7].

Code 4.55: Specific enthalpy of formation

```
        ifl hf298Method == lhvBased then
fhf1:   hf298 = lhv + wC*(-393510.0/12.011) + wH
   *(-241826.0/2.01588) +          wS*(-296835.0/32.066);
        elsel
fhf2:   if SubstNr == 10.0 then
        hf298 = 1197.99 - 562.28;
        else
        hf298 = 0.0;
endifl
```

Based on the ultimate analysis of the substance and the calculated lhv-value, the specific enthalpy of formation can be obtained according to Code 4.55, if the option *lhvbased* has been chosen. If not, the enthalpy of formation for tar is calculated through values from [7]. 562.28 kJ/kg is included for solid-gas state transition. For all the other substances *hf298* is set to 0 kJ/kg.

Code 4.56: Chemical exergy

```
        ifl ModelSubstance == Tar then
fe_chem_tar:    e_chem = 40331.5;
```

```
        endifl
        ifl ModelSubstance != Tar then
fe_chem:        e_chem = hhv;
        endifl
```

The specific exergy of the tar mixture -at the conditions of 25°C, 1 bar- is referred to [9], Subsection 5.5.6.

Solid

wAsh	Mass fraction Ash in total solid matter [kg/kg]
wK2O	Mass fraction K_2O in total solid matter [kg/kg]
wMgO	Mass fraction MgO in total solid matter [kg/kg]
wCaO	Mass fraction CaO in total solid matter [kg/kg]
wSiO2	Mass fraction SiO_2 in total solid matter [kg/kg]
wOlivine	Mass fraction Olivine in total solid matter [kg/kg]
wCaCO3	Mass fraction $CaCO_3$ in total solid matter [kg/kg]
wDolomite	Mass fraction Dolomite in total solid matter [kg/kg]
wCaSO4	Mass fraction $CaSO_4$ in total solid matter [kg/kg]
wCaOH2	Mass fraction $CaOH_2$ in total solid matter [kg/kg]
rho	Densitiy of the solid mixture [kg/m^3]
e_chem	Specific exergy at environmental conditions (25°C, 1 bar) [kJ/kg]
Functions	**Description**
sfht	Function returns the specific enthalpy (sensible heat only, h(0.01°C) = 0.0) of a solid mixture defined by solid stream composition [kJ/kg]
sfst	Function returns the specific entropy of a solid mixture defined by solid stream composition [kJ/kgK]
sfhf273	Function returns the specific enthalpy of formation of a solid mixture defined by solid stream composition at 273.15 K [kJ/kg]
sfhf278	Function returns the specific enthalpy of formation of a solid mixture defined by solid stream composition at 278.15 K [kJ/kg]

Table 4.10: Solid (inorganic) global item description

Code 4.57: Sum of mass fractions

```
fmass:  wAsh + wK2O + wMgO + wCaO + wSiO2 + wOlivine + wCaCO3 +
    wDolomite + wCaSO4 + wCaOH2 = 1.0;
```

Code 4.58: Mass specific exergy

```
fe_chem:        e_chem  = (wAsh*0.1735 + wK2O)*5246.49
                + wMgO*3019.97
                + (wAsh*0.7173 + wCaO)*3387.56
                + (wAsh*0.1092 + wSiO2)*28.52
                + wOlivine*(0.5*1290.79 + 0.5*1075.26)
                + wCaCO3*751.17
                + wDolomite*767.89
                + wCaSO4*79.11
                + wCaOH2*2026.80;
```

Like in the previous section, the mass specific exergy of solid mixtures at 25°C, 1 bar is based on the thermodynamic equilibrium according to [9], Subsection 5.5.6. The chemical exergy of a chemical compound is the energy that can be obtained by reversible transformation into environment compounds and reversible dilution to environmental concentration.

Code 4.59: Density of the solid mixture

```
frho:   1.0/rho = wAsh/3058.0 + wK2O/2320.0 + wMgO/3576.0 + wCaO
    /3400.0 + wSiO2/2648.0 + wOlivine/3733.0 + wCaCO3/2710.0 +
    wDolomite/3208.0 + wCaSO4/2960.0 + wCaOH2/2230.0;
```

The coefficients for density calculation of solid mixtures are taken from DAns Lax Data - Issue 04/Juni 2003.

4.2.3 Model Units

In the following part of the model library description, all default models of the various units are described.
In connection with this dissertation some modifications on already existing models had to be done. Additionally, some of them had to be created from the base. All changes in the existing library are summarized in Table 4.11.
The content of the BG_Lib library is continuously in progress. The actual state, described in this section, is named BG_Lib_2015_06 according to the year and month the last changes were been done.
Not all of these units have been used for this dissertation's simulations. However, this documentation and BG_Lib version will be the basis for further library modifications and simulations.

Model unit	Changes
Equilibrium unit for CO/CO_2 methation	An equilibrium unit containing the WGS reaction and CO methanation reaction was extended by the CO_2 methanation reaction
Methanation reactor	An already existing reactor for general application was adopted to meet the specific requirements for the simulation of the methanation process. This reactor is now available as an isothermal (with cooling) and an adiabatic one. For isothermal application gaseous, water and thermal oil cooling is now available
Condenser unit	The existing condenser unit did not contain the cooling water flow. All balances were changed to include the calculation of the cooling water feed or drain temperature and phase state
Membrane	The model of the membrane unit was based on permeabilities, it was change to be able to use the more common permeances of the various species
PEM-electrolyser	The model was is completely new. Due to its structure it can also be applied for alkaline and any other electrolysis technology
DLL-file	**Changes**
SimPropMeth	The original SimProp.dll database was extended by the CO_2 methanation reaction

Table 4.11: Summary of changes in the BG_Lib library

At the end of any model unit, except the few ones it is not reasonable, an energy and mass balance is illustrated. Both balances should proof that the unit is working properly if the input equals the output quantity. For both balance figures, on the left ordinate values at the same scale for in- and output are shown. The secondary ordinate, on the right side, shows the values for the difference between in- and output. Mostly the scale of the secondary ordinate differs from the first one as the difference is very small. This difference in scale could might cause the impression of a relevant difference between in- and output. However, for all illustrated energy and mass balances no significant difference has been investigated.

A difference is either caused by round off errors or inaccuracies due to an iteration accuracy of 10^{-6} and a maximum of 500 iteration steps.

Table 4.12 shows where the energy and mass balance values are taken from. Some of them are taken from the process simulations of Section 3.1 (I) and Section 3.2 (II). The other units were simulated one by one on their own for independent chosen values (III), just to proof the correctness of their energy and mass balance. Some of the balances are based on the original Oberwart simulation of calendar week 45 (IV).

All equilibrium models (u_chemeq_g_) are validated through an comparison of respective data calculated by HSC 6.0 instead of an energy and mass balance and therefore not listed in Table 4.12.

Model unit	I	II	III	IV
Column Gas-Water	x			
Combustion gas stream	x			
Combustion chamber	x			
Combustion-afterburner				x
Compressor gas	x			
PEM-Electrolyser	x			
Methanation reactor	x			
Condenser	x			
Membrane		x		
Evaporative cooler	x			
Flue gas denitrification			x	
Drum for liquid/steam separation			x	
Dryer	x			
Electric heater			x	
Fuel cell			x	
Gas engine				x
DFB gasifier	x			
Generator				x
Heat exchanger gas-gas	x			
Hydraulic switch			x	
Info unit gas	x			
Injector	x			
Loop connector	x			
Mixer gas	x			
Electric motor	x			
ORC				x
Gas oxidation reactor			x	
Pipe	x			
Pump	x			
Externally heated pyrolyzer			x	
Water quench			x	

Model unit	I	II	III	IV
Saturation temperature and pressure	x			
Organic scrubber	x			
PSA			x	
Separation organic-organic			x	
Separation inorganic-organic			x	
Separation water-organic			x	
Separation water-emulsion-organic	x			
Separation inorganic-gas	x			
Splitter gas	x			
Thermoelectric generator gas-gas			x	
Transformer steam-gas	x			
Trap			x	
Turbine gas				x
Valve gas	x			

Table 4.12: Source of balance figures

Ambient block u_amb_air_

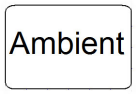


Figure 4.5: Ambient block library icon

For various unit operations it is relevant to know the ambient conditions in e.g. for the evaporative cooler when ambient air is drawn into the unit. Within the ambient block, the name of the ambient global can be defined as well as its properties (*t*, *p*, *h*, *phi_rel*) (Code 4.60 - 4.63) .

Variables	Description
t	Ambient temperature used for ambient air [°C]. This ambient temperature does not affect the values for exergy, where always 25°C are used as the reference temperature
p	Ambient air pressure correlated to the geodetic altitude [bar]
h	Altitude above sea level [m]
v	Specific volume [m^3/kg]
phi_rel	Pressure of H_2 product stream [bar]
N2	Volume fraction of N_2 in dry air
O2	Volume fraction of O_2 in dry air
Ar	Volume fraction of Ar in dry air
CO2	Volume fraction of CO_2 in dry air
Function	**Description**
wfp0t	Function returns the vapour pressure of water for a certain temperature calculated according to an Extended Antoine Equation
Global	**Description**
Ambient	Geographical and meteorological ambient conditions (referencing to ambient global)
AirComp	Gas composition of ambient air (referencing to gas global)

Table 4.13: Ambient block item description

Code 4.60: Ambient temperature

```
ft:      t = Ambient.t;
```

Code 4.61: Ambient pressure

```
fp:      p = Ambient.p;
```

Code 4.62: Geodetic altitude

```
fh:      h = Ambient.h;
```

Code 4.63: Relative humidity in ambient air at ambient conditions

```
fphi_rel:        phi_rel = Ambient.phi_rel;
```

Code 4.64: Specific volume of the carrier gas stream

```
fv:      v * AirComp.M * p * 100.0 = 8.31451 * (t + 273.15);
```

The specific gas volume *v* is calculated by Code 4.64. The equation is based on the ideal gas equation where M is the molar mass of the gas mixture and R the ideal gas constant.

Code 4.65: Mole fraction of H_2O in air

```
fyH2O: Ambient.p*AirComp.yH2O=(Ambient.phi_rel/100.0)*wfp0t(
    Ambient.t);
```

Via Code 4.65 the mole fraction of water -within the ambient air- can be calculated if the relative humidity is set or vice versa. The vapour pressure *wfp0t* of water -for a certain temperature- is calculated external within the SimProp DLL file, according to an extended Antoine Equation.

Code 4.66: Mole fractions of the AirComp

```
fyAr:    100.0*AirComp.yAr = (1.0 - AirComp.yH2O)*Ar;
fyC2H4:  AirComp.yC2H4 = 0.0;
                                    .
                                    .
fyCO:    AirComp.yCO = 0.0;
fyCO2:   1.0e+6*AirComp.yCO2 = (1.0 - AirComp.yH2O)*CO2;
                                    .
                                    .
fyN2:    100.0*AirComp.yN2 = (1.0 - AirComp.yH2O)*N2;
                                    .
                                    .
fyO2:    100.0*AirComp.yO2 = (1.0 - AirComp.yH2O)*O2;
fySO2:   AirComp.ySO2 = 0.0;
```

The composition of the ambient air (Code 4.66) is defined on dry base. It is assumed that it just contains the four species N_2, O_2, Ar and CO_2. All the other species are set to zero. The CO_2 share is defined in parts per million (ppm).

Chemical equilibrium models u_chemeq_g_

Theoretical information on the chemical reactors and chemical equilibrium is provided by T. Pröll in German [76][Section 5.3] and English [104].

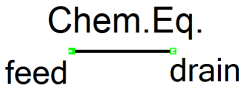


Figure 4.6: Equilibrium library icon

Chemical equilibrium for gases - Model methanation chemeq_meth_app3

Switch	Description
calc_eq_CO_meth	Activate or deactivate the equilibrium calculation of CO
calc_WGS_CO_meth	Activate or deactivate the equilibrium calculation of the WGS reaction
calc_eq_CO2_met	Activate or deactivate the equilibrium calculation of CO_2
Variables	**Description**
pd_eq_CO_meth	Logarithmic distance from CO-methanation equilibrium [-]
Kp_CO_meth	Partial pressure equilibrium constant for the CO-methanation reaction [-]
pd_eq_WGS	Logarithmic distance from WGS equilibrium [-]
Kp_WGS	Partial pressure equilibrium constant for the WGS reaction [-]
pd_eq_CO2_meth	Logarithmic distance from CO_2-methanation equilibrium [-]
Kp_CO2_meth	Partial pressure equilibrium constant for the CO_2-methanation reaction [-]
Functions	**Description**
Kp_CO_H2O_CO2_H2	Function returns equilibrium constant for partial pressures of the water gas shift reaction, its reciprocal value is used for the methanation application
Kp_CH4_H2O_CO_3H2	Function returns the equilibrium constant for partial pressures of the CH4 reforming reaction $CH_4 + H_2O \rightarrow CO + 3\ H_2$, its reciprocal value is used for the methanation application
Kp_CO2_4H2_CH4_2H2O	Function returns equilibrium constant for partial pressures of the CO_2 methanation reaction

Table 4.14: Chemical equilibrium model for methanation, item description

First of all it is important to clarify that the model does not converge if all three equations (CO-methanation, WGS-reaction, CO_2-methanation) are activated through the switch. It is a general problem that just two equilibrium equations can be considered due to convergence issues. If the CO-methanation and the WGS equation are executed, the CO_2-methanation is considered as well due to its linear relation to the other two ones.

Code 4.67: Temperature equality

```
ft:      drain.t = feed.t;
```

Code 4.68: Pressure equality

```
fp:      feed.p = drain.p;
```

Through the equations of Code 4.67 and 4.68, the outgoing temperature and pressure are equal to the incoming.

Code 4.69: Mass flow considerations for the various substances

```
# Gas
fmassflow:       drain.massflow = feed.massflow;
# Dust   ifl (ref(feed.Dust) && ref(drain.Dust)) then
         fmass_dust:      drain.dust_content = feed.dust_content;
         endifl
# Char   ifl ref(feed.Char) && ref(drain.Char) then
         fmass_char:      drain.char_content = feed.char_content;
         endifl
# Tar    ifl ref(feed.Tar) && ref(drain.Tar) then
         fmass_tar:       drain.tar_content = feed.tar_content;
         endifl
```

Via Code 4.69 the statement can be done, that the mass flow for all substances is constant if they are referenced.
The following code lines are the core elements of the equilibrium unit.

Code 4.70: CO methanation

```
ifl (calc_eq_CO_meth == yes) then
f_Kp_CO:         1 = Kp_CH4_H2O_CO_3H2(drain.t) * Kp_CO_meth;

f_pd_eq_CO_meth: if isconverged(pd_eq_CO_meth) then     10^(
    pd_eq_CO_meth)*(Kp_CO_meth*drain.p^(-2)) = (drain.Gas.yCH4*
    drain.Gas.yH2O)/(drain.Gas.yCO*(drain.Gas.yH2)^3);
else     pd_eq_CO_meth + log(Kp_CO_meth*drain.p^(-2)) = log((drain.
    Gas.yCH4*drain.Gas.yH2O)/(drain.Gas.yCO*(drain.Gas.yH2)^3));
elsel    frpd_CO_meth: pd_eq_WGS = -1000;
endifl
```

Code 4.71: Water-gas shift equilibrium

```
ifl (calc_eq_WGS == yes) then
f_kp_WGS:        1 = Kp_CO_H2O_CO2_H2(drain.t) * Kp_WGS;
frpd_WGS:        if isconverged(pd_eq_WGS)       then     10^(
    pd_eq_WGS)* Kp_WGS = (drain.Gas.yCO*drain.Gas.yH2O)/(drain.Gas.
    yCO2*drain.Gas.yH2);
else     pd_eq_WGS + log(Kp_WGS) = log((drain.Gas.yCO*drain.Gas.
    yH2O)/(drain.Gas.yCO2*drain.Gas.yH2));
elsel frpd_WGS_2: pd_eq_WGS = -1000;
endifl
```

Code 4.72: CO_2-methanation

```
ifl (calc_eq_CO2_met == yes) then
f_Kp:    Kp_CO2_meth = Kp_CO2_4H2_CH4_2H2O(drain.t);
frpd_eq_CO2_meth: if isconverged(pd_eq_CO2_meth)
then 10^(pd_eq_CO2_meth)*Kp_CO2_4H2_CH4_2H2O(drain.t)*drain.p^(2)
    = drain.Gas.yCH4*drain.Gas.yH2O^2/(drain.Gas.yCO2*drain.Gas.yH2
    ^4);
else pd_eq_CO2_meth + log(Kp_CO2_4H2_CH4_2H2O(drain.t)*drain.p^(2)
    ) = log(drain.Gas.yCH4*(drain.Gas.yH2O)^2/(drain.Gas.yCO2*(
    drain.Gas.yH2)^4));
elsel
frpd_CH4_2: pd_eq_CO2_meth = -1000;
endifl
```

$$10^{\delta} \cdot K_p = \frac{p_{CH_4} \cdot p_{H_2O}^2}{p_{CO_2} \cdot p_{H2}^4} \tag{4.2}$$

$$p_i = y_i \cdot p \tag{4.3}$$

In order to include the CO_2-methanation reaction to the SimProp.dll property database, the database was first duplicated and further extended by Equation 2.21. The new database is named SimPropMeth.dll. Both DLL-files are using NASA polynomials [27] to gain property data of the various substances. The procedure is well described in the dissertation of M. Stidl [102] and T. Pröll [76].
Code 4.70 to 4.72 are based on the concept of Equation 4.2. The partial pressure p_i is expressed after the law of Dalton (Eq.4.3) where p represents the total pressure and y_i the mole fraction. K_p is the equilibrium constant related to partial pressures. Assuming an ideal gas, K_p is just temperature depended. It is calculated in the external SimPropMeth.dll file and afterwords returned to IPSEpro. The term on the right side -of Equation 4.2- calculates the equilibrium constant by the actual drain gas composition. δ is the difference to full chemical equilibrium. For chemical equilibrium the left and the right term of the equation have be equal. Therefore, δ has to be zero. If δ is negative, the chemical equilibrium is on the side of reactants, if δ is positive it is on the products side. Actually, a reaction cannot react beyond chemical equilibrium. Therefore, δ larger zero points out a failure.
This approach can be adopted for any other chemical reaction defined in the simprop or SimPropMeth DLL-database. The so called radio button (switch-function) activates or deactivates the various equations. For example if just the WGS-reaction should be considered, the radio button "yes" is chosen for the WGS-reaction and "no" for all the other ones. The simulation results for the methanation units of Chapter 3 are simulated through the combination of an reactor unit and the equilibrium model for methanation. The reactor model provides all balances and state definitions while the equilibrium model considers the chemical reactions. The results IPSEpro results for CO_2-methanation (dots) are validated with data of HSC 6.0 (dashed lines), see Figure 4.7. The validation shows good agreement for the relevant temperature range.

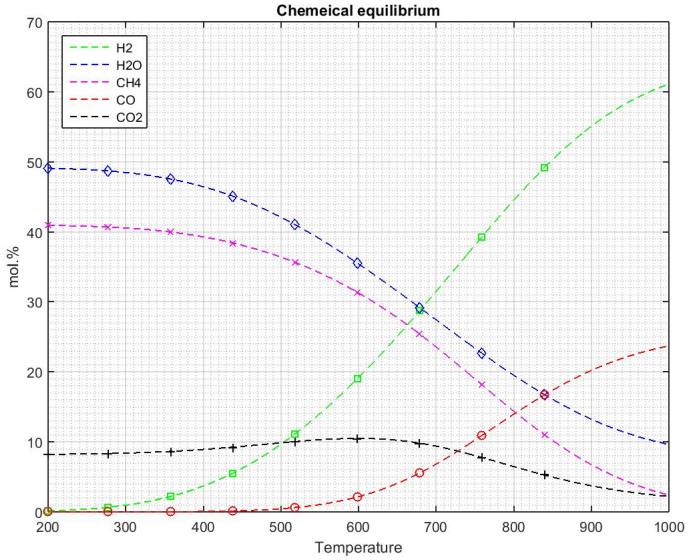


Figure 4.7: Methanation results validation at 2 bar

Chemical equilibrium for gases - Model oxidation chemeq_ox

Variables	Description
pd_eq_oxCO	Logarithmic distance from equilibrium for CO oxidation
Functions	**Description**
Kp_CO_O2_CO2	Returns the partial pressure equilibrium constant for the reaction CO + 0.5 O_2 → CO_2

Table 4.15: Chemical equilibrium model for oxidation, item description

Code 4.73: CO oxidation

```
frpd_oxCO:      if isconverged(pd_eq_oxCO)      then
 10^(pd_eq_oxCO)*(Kp_CO_O2_CO2(drain.t)*drain.p^0.5) = drain.Gas.
    yCO2/(drain.Gas.yCO*(drain.Gas.yO2)^0.5);
else     pd_eq_oxCO + log(Kp_CO_O2_CO2(drain.t)*drain.p^0.5) = log(
   drain.Gas.yCO2/(drain.Gas.yCO*(drain.Gas.yO2)^0.5));
```

This model calculates the equilibrium for CO oxidation (see Table 4.15). In the model source code, Code 4.73, the mole fraction y_i is replaced by the law of Dalton (Eq.4.3) equal to the CO_2-methanation model. Also distance to equilibrium, mass balance, temperature and pressure are defined equal.

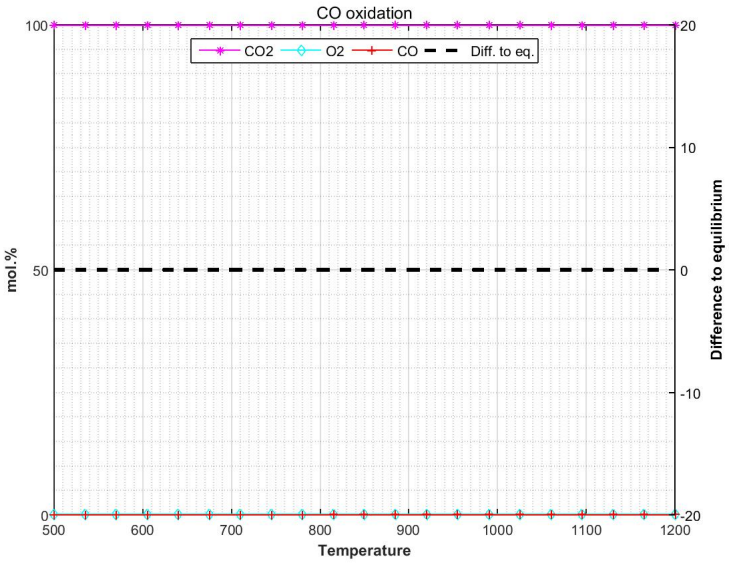


Figure 4.8: CO oxidation model validation at 1 bar

Figure 4.8 shows a good fit for the relevant temperature range between the IPSEpro equilibrium values compared to data from HSC 6.0. The shown validation procedure is applied for the equilibrium reactions of Figure 4.9 as well. In HSC 6.0 the chemical equilibrium composition of the various equations is calculated, based on the reaction's stoichiometry. For each different reaction and temperature a stream was defined in IPSEpro with the relating equilibrium composition calculated in HSC 6.0. These streams were connected to a chemical equilibrium model (*u_chemeq_g_*) and the respective unit-model and reaction was chosen. In case of CO oxidation the *chemeq_ox* model was chosen. As the model just contains the one equation of CO oxidation (Code 4.73) the respective equation did not have to be picked separately. After running the simulation and importing the results, the difference to chemical equilibrium -calculated by IPSEpro- can be seen by the chemical equilibrium unit. The difference to chemical equilibrium should be zero, as described in the previous Subsection 4.2.3.

Chemical equilibrium for gases - Model reduction chemeq_red

This is the standard model for chemical equilibrium in reducing atmosphere. Dust, char and tar are optionally possible in the gas stream. All composition objects (globals) must be the same in *drain* and *feed* stream.

Switch	Description
calc_eq_WGS	Activate or deactivate the equilibrium calculation of WGS
calc_eq_CH4	Activate or deactivate the equilibrium calculation of CH_4
calc_eq_C2H4	Activate or deactivate the equilibrium calculation of C_2H_4
calc_eq_C2H6	Activate or deactivate the equilibrium calculation of C_2H_6
calc_eq_C3H8	Activate or deactivate the equilibrium calculation of C_3H_8
calc_eq_NH3	Activate or deactivate the equilibrium calculation of NH_3
calc_eq_HCN	Activate or deactivate the equilibrium calculation of HCN
calc_eq_oxCO	Activate or deactivate the equilibrium calculation of oxCO
Variables	**Description**
pd_eq_WGS	Logarithmic distance from WGS equilibrium [-]
pd_eq_CH4	Logarithmic distance from methane reforming equilibrium [-]
pd_eq_C2H4	Logarithmic distance from ethene reforming equilibrium [-]
pd_eq_C2H6	Logarithmic distance from ethane reforming equilibrium [-]
pd_eq_C3H8	Logarithmic distance from propane reforming equilibrium [-]
pd_eq_NH3	Logarithmic distance from ammonia synthesis equilibrium [-]
pd_eq_HCN	Logarithmic distance from hydrogen cyanide reforming equilibrium [-]
pd_eq_oxCO	Logarithmic distance from CO oxidation equilibrium [-]
Functions	**Description**
Kp_CO_H2O_CO2_H2	Function returns equilibrium constant for partial pressures of the water gas shift reaction CO + $H_2O \leftrightarrow H_2 + CO_2$
Kp_CH4_H2O_CO_3H2	Function returns the equilibrium constant for partial pressures of the CH4 reforming reaction $CH_4 + H_2O \leftrightarrow CO + 3\ H_2$
Kp_C2H4_2H2O_2CO_4H2	Returns the partial pressure equilibrium constant for the reaction $C_2H_4 + 2H_2O \leftrightarrow 2CO + 4H_2$
Kp_C2H6_2H2O_2CO_5H2	Returns the partial pressure equilibrium constant for the reaction $C_2H_6 + 2H_2O \leftrightarrow 2CO + 5H_2$
Kp_C3H8_3H2O_3CO_7H2	Returns the partial pressure equilibrium constant for the reaction $C_3H_8 + 3H_2O \leftrightarrow 3CO + 7H_2$
Kp_N2_3H2_2NH3	Returns the partial pressure equilibrium constant for the ammonia synthesis reaction $N_2 + 3H_2 \leftrightarrow 2NH_3$
Kp_NH3_CO_HCN_H2O	Returns the partial pressure equilibrium constant of the reaction $NH_3 + CO \leftrightarrow HCN + H_2O$
Kp_CO_O2_CO2	Returns the partial pressure equilibrium constant for the reaction CO + 0.5 $O_2 \leftrightarrow CO_2$

Table 4.16: Chemical equilibrium model for reduction, item description

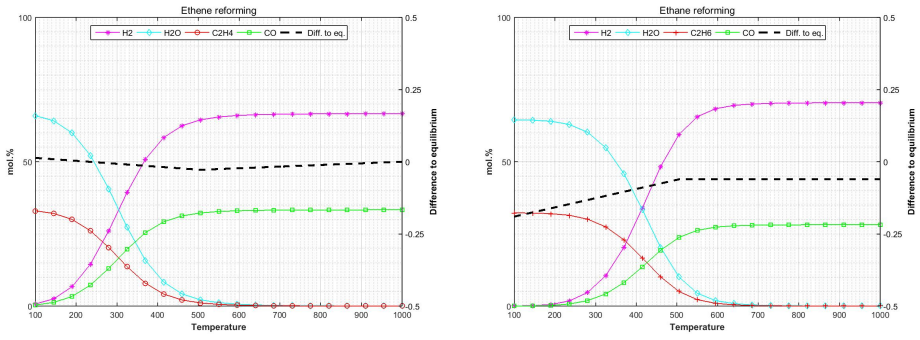
Code 4.74: Reforming of methane

```
ifl (calc_eq_CH4 == yes) then
frpd_CH4: if isconverged(pd_eq_CH4) then 10^(
    pd_eq_CH4)*(Kp_CH4_H2O_CO_3H2(drain.t)*drain.p^(-2)) = drain.
    Gas.yCO*(drain.Gas.yH2)^3/(drain.Gas.yCH4*drain.Gas.yH2O);
          else pd_eq_CH4 + log(Kp_CH4_H2O_CO_3H2(drain.t)
              *drain.p^(-2)) = log(drain.Gas.yCO*(drain.Gas.
              yH2)^3/(drain.Gas.yCH4*drain.Gas.yH2O));
elsel
frpd_CH4_2: pd_eq_CH4 = -1000;
endifl
```

Code 4.75: Reforming of ethene

```
ifl (calc_eq_C2H4 == yes) then
frpd_C2H4: if isconverged(pd_eq_C2H4) then 10^(
    pd_eq_C2H4)*(Kp_C2H4_2H2O_2CO_4H2(drain.t)*drain.p^(-3)) = (
    drain.Gas.yCO)^2*(drain.Gas.yH2)^4/(drain.Gas.yC2H4*(drain.Gas.
    yH2O)^2);
          else pd_eq_C2H4 + log(Kp_C2H4_2H2O_2CO_4H2(
              drain.t)*drain.p^(-3)) = log((drain.Gas.yCO)
              ^2*(drain.Gas.yH2)^4/(drain.Gas.yC2H4*(drain.
              Gas.yH2O)^2));
elsel
frpd_C2H4_2: pd_eq_C2H4 = -1000;
endifl
                                    .
                                    .
```

Some of the equilibrium equations are considered in several equilibrium models. The calculation of *pd_eq_WGS*, the logarithmic distance from WGS equilibrium, can be seen for the methanation model, Code 4.71. For each and every equilibrium reaction of Table 4.16 the same procedure is applied. Every difference to full equilibrium is defined by its own δ (see Eq.4.2) as presented in the previous two sections. Therefore, just five of them are presented here.



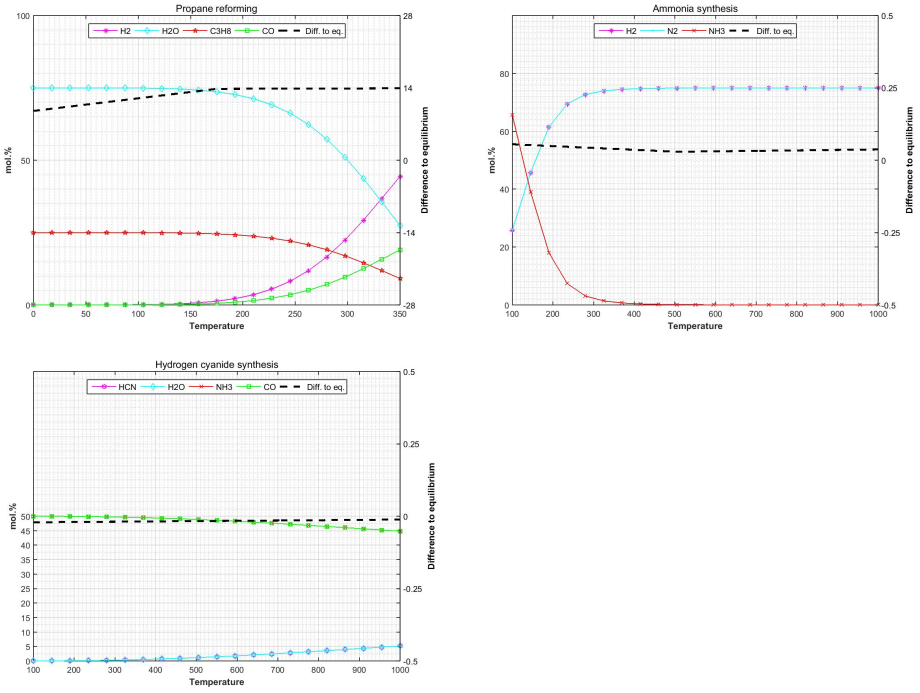


Figure 4.9: Equilibrium composition for the equations of the reduction unit

An error has been found for the propane steam reforming equation (Eq.4.4).

$$C_3H_8 + 3H_2O \rightleftharpoons 3CO + 7H_2 \tag{4.4}$$

The equilibrium constant K_p e.g. of Equation 4.2 is calculated through the Gibbs free enthalpy (G) (Eq.4.5).

$$ln(K_p) = -\frac{\Delta G_R^0(T)}{R \cdot T} \tag{4.5}$$

For an ideal gas it is assumed that the Gibbs free enthalpy is only dependent on the temperature. According to [104], "equilibrium is reached where any further change in composition due to chemical reactions would lead to an increase of the intensive Gibbs free enthalpy". In other words, equilibrium is reached when the Gibbs free enthalpy is at its minimum (Eq. 4.6).

$$G = H^* - T \cdot S \rightarrow Min \tag{4.6}$$

In order to find the error of the propane steam reforming equation, $G_R^0(T)$ has been calculated through HSC 6.0 and IPSEpro (NASA polynomial). Figure 4.10 shows the relative error when comparing the calculated values.

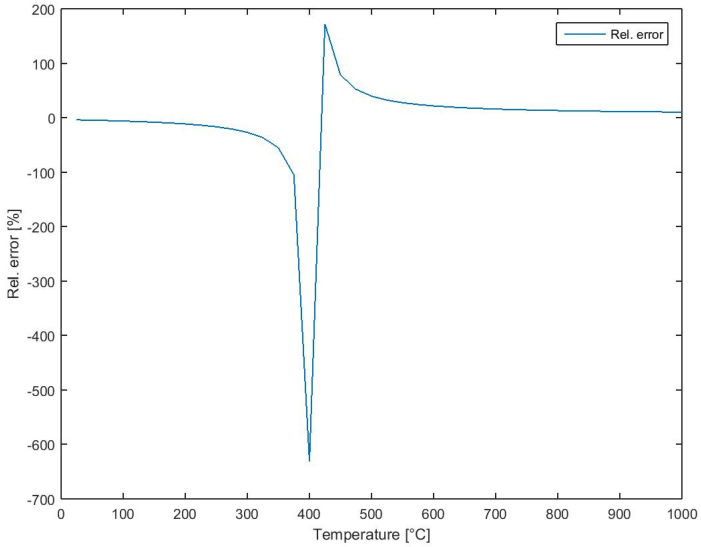


Figure 4.10: Relative error of $G_R^0(T)$ at different temperatures

Therefore, it has been investigated that the equilibrium deviation for propane steam reforming in Figure 4.9 is either related to false values from HSC 6.0 or more likely to wrong coefficients for the NASA polynomial. For all the other equations no error occur.

Column Gas-Water u_col_gw_

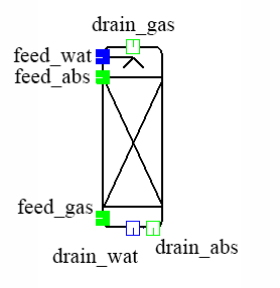


Figure 4.11: Column gas-water library icon

Figure 4.11 shows the connectors of the column unit. At the entrance *feed_abs* the absorbent enters, it leaves through exit *drain_abs*. The loaded gas stream enters at the connection *feed_gas*. The cleaned gas stream leaves via the exit *drain_gas*. Quench water is sent into the column at *feed_wat* and leaves at *drain_wat*.

Variables	Description
dp_gas	Pressure drop for gas stream [bar]
dp_wat	Pressure drop for water stream [bar]
dp_drain	Pressure difference between water and gas stream exiting the quench [bar]
dt_top	Temperature difference between exiting gas and water feed [°C]
dt_bot	Temperature difference between exiting water and gas feed [°C]
H_abs_CO2	Specific enthalpy of CO_2 absorption (average value for the whole column) [kJ/mol]
Q_abs	Heat release because of transition of state (absorption positive, desorption negative) [kW]
E_loss	Exergy loss in the quench [kW]
m_CO2_feed	Mole fraction of CO_2 in water feed/absorbed gas feed [mol/kg]
m_CO2_drain	Mole fraction of CO_2 in water drain/absorbed gas drain [mol/kg]
pd_eq_CO2_top	Logarithmic distance to Henry equilibrium in the column top (gas exit) [-]
pd_eq_CO2_bot	Logarithmic distance to Henry equilibrium in the column bottom (gas feed) [-]
lgH_CO2_top	Logarithmic distance from CO oxidation equilibrium [$\frac{bar \cdot kg}{mol}$]
lgH_CO2_bot	Logarithmic distance from CO oxidation equilibrium [$\frac{bar \cdot kg}{mol}$]

Table 4.17: Column gas-water item description

Code 4.76: Energy balance

```
fEnergy:        drain_gas.massflow*drain_gas.h_total + drain_wat.
    massflow*drain_wat.h_total + drain_abs.massflow*drain_abs.
    h_total = 3600.0*Q_abs + feed_gas.massflow*feed_gas.h_total +
    feed_wat.massflow*feed_wat.h_total + feed_abs.massflow*feed_abs
    .h_total;
```

Code 4.77: Mass balance H_2O

```
fmass_H2O:      drain_gas.massflow*drain_gas.Gas.wH2O + drain_wat.
    massflow + drain_abs.massflow*drain_abs.Gas.wH2O = feed_gas.
    massflow*feed_gas.Gas.wH2O + feed_wat.massflow + feed_abs.
    massflow*feed_abs.Gas.wH2O;
```

Code 4.77 shows the overall mass balance for water within the unit. For all the other species -considered in IPSEpro- their mass balance is equally defined to the balance of Code 4.77.

Code 4.78: Species mass balances

```
fmass_Ar:       drain_gas.massflow*drain_gas.Gas.wAr
                + drain_abs.massflow*drain_abs.Gas.wAr
                = feed_gas.massflow*feed_gas.Gas.wAr + feed_abs.
                    massflow*feed_abs.Gas.wAr;
fmass_C2H4:     drain_gas.massflow*drain_gas.Gas.wC2H4
                + drain_abs.massflow*drain_abs.Gas.wC2H4
                = feed_gas.massflow*feed_gas.Gas.wC2H4 + feed_abs.
                    massflow*feed_abs.Gas.wC2H4;
                                .
                                .
                                .
fmass_O2:       drain_gas.massflow*drain_gas.Gas.wO2
                + drain_abs.massflow*drain_abs.Gas.wO2
                = feed_gas.massflow*feed_gas.Gas.wO2 + feed_abs.
                    massflow*feed_abs.Gas.wO2;
fmass_SO2:      drain_gas.massflow*drain_gas.Gas.wSO2
                + drain_abs.massflow*drain_abs.Gas.wSO2
                = feed_gas.massflow*feed_gas.Gas.wSO2 + feed_abs.
                    massflow*feed_abs.Gas.wSO2;
```

The species balance for each and every on of the gaseous species is defined after Code 4.78. It is assumed that no gaseous species gets into the water stream. Therefore, water is not considered in the mass balance of any gaseous species. Further, in Code 4.84 the water content of the drain absorbent stream is set to zero.

Code 4.79: Pressure drops

```
fdp_gas:        drain_gas.p + dp_gas = feed_gas.p;
fdp_wat:        drain_wat.p + dp_wat = feed_wat.p;
```

dp_gas and dp_wat are calculated after the equations of Code 4.79.

Code 4.80: Pressure difference between exiting gas and exiting water

```
fdp_drain:        drain_gas.p + dp_drain = drain_wat.p;
```

Code 4.80 presents the pressure difference between exiting gas and exiting water stream.

Code 4.81: Temperature difference between gas drain and water feed (column top)

```
fdt_bot:          feed_gas.t = drain_wat.t + dt_bot;
```

Code 4.82: Temperature difference between gas feed and water drain (column bottom)

```
fdt_top:          drain_gas.t = feed_wat.t + dt_top;
```

The temperature differences at the top and the bottom of the column, between the gas and water streams, are calculated after Code 4.81 and 4.82.

Code 4.83: Pressure and temperature in absorbed gas stream

```
fpdrain_abs: drain_abs.p = drain_wat.p;
ftdrain_abs: drain_abs.t = drain_wat.t;
```

Code 4.83 defines the pressure and the temperature for the absorbent stream.

Code 4.84: Zero H_2O content in adsorbed gas drain

```
fdrain_abs_H2O: drain_abs.Gas.yH2O = 0.0;
```

Code 4.85: Exergy loss

```
fE_loss:          drain_gas.Exergy + drain_wat.Exergy + drain_abs.
    Exergy + E_loss = feed_gas.Exergy + feed_wat.Exergy + feed_abs.
    Exergy;
```

Code 4.86: Relative humidity always 100% in exiting gas stream

```
fphirel:          drain_gas.Gas.yH2O*drain_gas.p = feed_wat.wfp0t(
    drain_gas.t);
```

It is assumed that the relative humidity is always 100% (φ=1) in the exiting gas stream (see comment MDK source code).

$$\varphi = \frac{actual\ vapour\ pressure}{saturation\ vapour\ pressure} \tag{4.7}$$

Due to the definition of the relative humidity (Eq.4.7), the partial pressure of steam equals the saturated vapor pressure (Code 4.86). The saturated vapour pressure is calculated -in the SimProp.dll file- by an extended Antoine equation and further returned to IPSEpro.

Code 4.87: Absorption thermal effects

```
fQ_abs: 3.6*Q_abs = (drain_abs.massflow*drain_abs.Gas.wCO2 -
    feed_abs.massflow*feed_abs.Gas.wCO2)*H_abs_CO2/44.0098;
```

The heat release through state transition can be calculated via Code 4.87 if *H_abs_CO2* -the specific enthalpy of CO_2 absorption- is known or vice versa. The factor 3.6 results from the unit conversion of kg/h into kg/s and the conversion of the mole mass of CO_2 from 44.0098 g/mol into kg/mol.

Code 4.88: Liquid phase mole fractions

```
fm_CO2_feed:    m_CO2_feed*feed_wat.massflow    = feed_abs.
    massflow*feed_abs.Gas.wCO2/44.0098e-3;
fm_CO2_drain:   m_CO2_drain*drain_wat.massflow  = drain_abs.
    massflow*drain_abs.Gas.wCO2/44.0098e-3;
```

Code 4.88 calculates the CO_2 liquid phase mole fractions. The mole mass of CO_2 is inserted in kg/mol.

Code 4.89: Henry constant for CO_2 in water

```
flgH_CO2_top:   lgH_CO2_top = -108.3808 - 0.01985076*(feed_wat.t +
     273.15) + 40.45154*log(feed_wat.t + 273.15) + 6919.53/(
    feed_wat.t + 273.15) - 669365.0/((feed_wat.t + 273.15)^2);
flgH_CO2_bot:   lgH_CO2_bot = -108.3808 - 0.01985076*(drain_wat.t
    + 273.15) + 40.45154*log(drain_wat.t + 273.15) + 6919.53/(
    drain_wat.t + 273.15) - 669365.0/((drain_wat.t + 273.15)^2);
```

The equations of Code 4.89 are calculating the Henry constant for CO_2 in water. The Henry constant describes the gas absorption capacity of a liquid phase. In the end the gaseous and liquid phase are in a solution equilibrium. Therefore, [42] describes the Henry constant as equilibrium constant K. In this article more information about the temperature dependent equation approach of Code 4.89 and its coefficients can be found.

Code 4.90: Equilibrium distances applied for Henry-equation

```
fpd_eq_CO2_top: if isconverged(pd_eq_CO2_top) then 10^(-
    pd_eq_CO2_top)*(10^(lgH_CO2_top)*m_CO2_feed) = (drain_gas.p*
    drain_gas.Gas.yCO2);
                else - pd_eq_CO2_top + log(10^(lgH_CO2_top)*
                    m_CO2_feed) = log(drain_gas.p*drain_gas.Gas.
                    yCO2);

fpd_eq_CO2_bot: if isconverged(pd_eq_CO2_bot) then 10^(-
    pd_eq_CO2_bot)*(10^(lgH_CO2_bot)*m_CO2_drain) = (feed_gas.p*
    feed_gas.Gas.yCO2);
                else - pd_eq_CO2_bot + log(10^(lgH_CO2_bot)*
                    m_CO2_drain) = log(feed_gas.p*feed_gas.Gas.yCO2
                    );
```

For Code 4.90 the Henry law ($p_i = k_H \cdot c_i$) is considered. The partial pressure in the gaseous phase p_i is substituted by the total pressure p times the CO_2 mole fraction, in the gaseous feed phase. The CO_2 fraction in the absorbent c_i is represented by the mole fraction CO_2 in water (*m_CO2_feed/drain*). The Henry constant k_H is described via *lgH_CO2_top*.
As *lgH_CO2_top* is calculated as the maximum gas absorption capacity of a liquid phase (Code 4.89), *pd_eq_CO2_top* describes the difference from actual state to maximum state.

If the calculation of Code 4.90 does not converge, there is also a second version of both equations in logarithmic form.

Figure 4.12 pictures the energy and mass balance of the scrubber unit. Both balances are

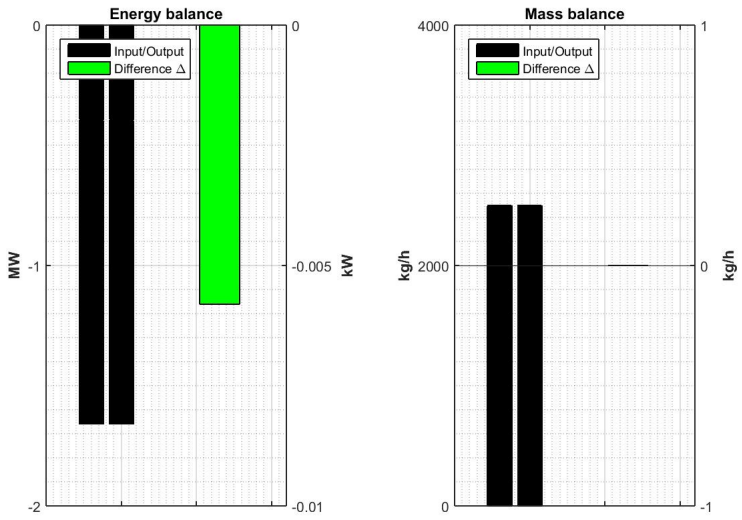


Figure 4.12: Energy and mass balance for the gas-water column

applied for the simulation of Section 3.1.

Composition gas stream unit u_comp_g_

Figure 4.13: Composition library icon

The composition of a stream can be defined by such an unit. The composition of a gas stream by its molar fraction is considered in the following. The same principle is applied for organic, solid and water streams. The unit is fed via the terminal *feed*, the gas stream leaves at the exit called *drain*. The items of Table 4.18 are defined through Code 4.91 to 4.95.

Variables	Description
Ar	Volume part of argon in the gas mixture [vol.%]
C2H4	Volume part of ethylene in gas mixture [vol.%]
C2H6	Volume part of ethane in the gas mixture [vol.%]
C3H8	Volume part of propane in the gas mixture [vol.%]
CH4	Volume part of methane in the gas mixture [vol.%]
CO	Volume part of carbon oxide in the gas mixture [vol.%]
CO2	Volume part of carbon dioxide in the gas mixture [vol.%]
H2	Volume part of hydrogen in the gas mixture [vol.%]
H2O	Volume part of water in the gas mixture [vol.%]
H2S	Volume part of hydrogen sulfide in the gas mixture [vol.%]
HCl	Volume part of hydrogen chloride in the gas mixture [vol.%]
HCN	Volume part of hydrogen cyanide in the gas mixture [vol.%]
N2	Volume part of nitrogen in the gas mixture [vol.%]
N2O	Volume part of nitrous oxide in the gas mixture [vol.%]
NH3	Volume part of ammonia in the gas mixture [vol.%]
NO	Volume part of nitrogen oxide in the gas mixture [vol.%]
O2	Volume part of oxygen in the gas mixture [vol.%]
SO2	Volume part of sulfur dioxide in the gas mixture [vol.%]
LHV	Lower heating value specific to gas volume at standard conditions (1.01325 bar, 273.15 K) [MJ/Nm^3]
HHV	Higher heating value specific to gas volume at standard conditions (1.01325 bar, 273.15 K) [MJ/Nm^3]

Table 4.18: Composition item description

Code 4.91: Equality of temperature

```
f_temp: drain.t = feed.t;
```

Code 4.92: Continuity of mass

```
f_mass: drain.massflow = feed.massflow;

ifl ref(feed.Dust) then f_dust_content: drain.dust_content = feed.
    dust_content; endifl
ifl ref(feed.Char) then f_char_content: drain.char_content = feed.
    char_content; endifl
ifl ref(feed.Tar) then  f_tar_content:  drain.tar_content = feed.
    tar_content;   endifl
```

Code 4.93: Equality of pressure

```
f_press:        drain.p = feed.p;
```

Code 4.94: Ideal gas relations for mole fraction

```
fAr:    Ar      = 100*feed.Gas.yAr;
fC2H4:  C2H4    = 100*feed.Gas.yC2H4;
fC2H6:  C2H6    = 100*feed.Gas.yC2H6;
fC3H8:  C3H8    = 100*feed.Gas.yC3H8;
fCH4:   CH4     = 100*feed.Gas.yCH4;
fCO:    CO      = 100*feed.Gas.yCO;
fCO2:   CO2     = 100*feed.Gas.yCO2;
fH2:    H2      = 100*feed.Gas.yH2;
fH2O:   H2O     = 100*feed.Gas.yH2O;
fH2S:   H2S     = 100*feed.Gas.yH2S;
fHCl:   HCl     = 100*feed.Gas.yHCl;
fHCN:   HCN     = 100*feed.Gas.yHCN;
fN2:    N2      = 100*feed.Gas.yN2;
fN2O:   N2O     = 100*feed.Gas.yN2O;
fNH3:   NH3     = 100*feed.Gas.yNH3;
fNO:    NO      = 100*feed.Gas.yNO;
fO2:    O2      = 100*feed.Gas.yO2;
fSO2:   SO2     = 100*feed.Gas.ySO2;
```

Code 4.95: Accesing the LHV and HHV value of the global

```
f_lhv:  LHV=feed.Gas.LHV;
f_hhv:  HHV=feed.Gas.HHV;
```

Combustion chamber u_comb_dfb01_fg

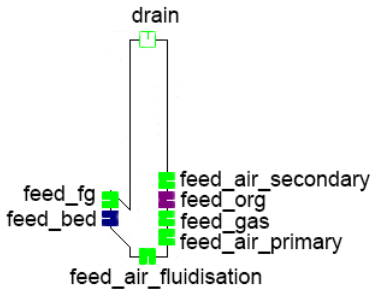


Figure 4.15: Combustion chamber library icon

In the following all input and output connections of Figure 4.15 are described. Entrance *feed_fg* provides a share of the oxidant, plus separated particles from the product gas stream, to the combustion chamber. For the simulation of Section 3.1 (DFB+PEM) instead of air, oxygen and flue gas are fed through all the fluidization entrances. Inlet *feed_bed*

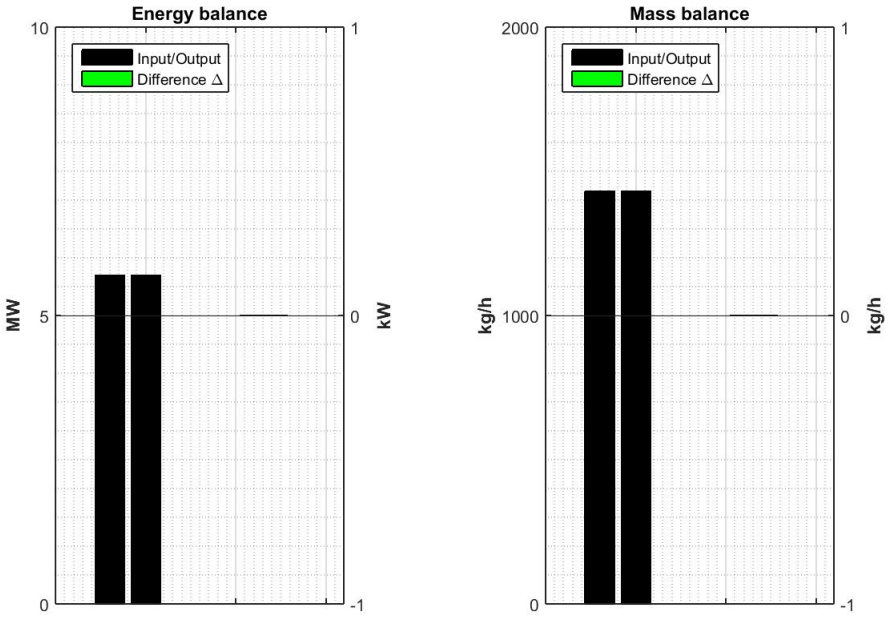


Figure 4.14: Energy and mass balance for the gas composition unit

is for solid bed material. Entrance *feed_air_fluidisation* also provides the oxidant for fluidization as well as inlet *feed_primary_air* and *feed_secondary_air*. The input *feed_gas* is installed to recycle a share of the product gas -when doing DFB-gasification- into the combustion chamber while *feed_org* is for loaded organic scrubbing media. For process simulations considering biomass gasification, the organic inlet stream is tar loaded RME from the product gas scrubber. The *drain* connection is the only outlet. The combination of a gasifier and combustion chamber (riser) unit are further described in detail by [76][5.2].

Switch	Description
NOx_formation	Model allows description of thermal NOx formation according to the switch position
Variables	**Description**
dp_riser	Total pressure drop in riser, from air manifold bottom to fg-exit [bar]
dp_fluid	Pressure drop from fluidisation air to loaded flue gas at top of riser [bar]
dp_air_prim	Pressure drop from primary air to loaded flue gas at top of riser [bar]
dp_air_sec	Pressure drop from primary air to loaded flue gas at top of riser [bar]
dp_gas	Pressure drop from fuel gas to loaded flue gas at top of riser [bar]
dp_org	Pressure drop from organic fuel to loaded flue gas at top of riser [bar]
dp_fg	Pressure drop from flue gas recirculation to loaded flue gas at the top of the riser [bar]
lambda	Air ratio of the combustion chamber based on the amount of oxygen in air stream needed for stoichiometric combustion of all substances entering the combustion chamber (e.g. also combustables entering with the air stream) [kg/kg]
CO_slip	Molar ratio between CO and CO_2 in flue gas. This value must be determined by the system (set or exiting CO concentration set) if lambda ≥ 1.0. If lambda < 1.0 the CO_slip has no influence on the system and must be set to a dummy value [mol/mol]
char_slip	Ratio between uncombusted char in drain gas and incoming char in feed gas. Caution: different units used for the char content in feed [kg/kg_inorg] and drain [g/Nm^3]. Additional information on the implementation of char_slip is given in [60][3.2.2.1]
P_therm	Thermal power of the combustion chamber based on lower heating values of all combustibles entering the reactor [kW]
q_loss_rel	Relative heat loss of the combustion reactor in % of thermal power [%]
Q_loss	Heat loss of the combustion reactor [kW]
E_loss	Exergy loss caused by the combustion process [kW]
A_riser	Effective sectional area of the riser. Important for the definition of the gas velocity in the riser [m^2]
w_riser	Superficial gas velocity in the riser [m/s]
Gs	Specific solid flow rate through riser: Gs = massflow_solid/A_riser [kg/m^{2}s]

Table 4.19: Combustion chamber item description

Code 4.96: Energy balance of combustion chamber

```
fEnergy: drain.massflow*drain.h_total + Q_loss*3600.0 = feed_bed.
    massflow*feed_bed.h_total + feed_fg.massflow*feed_fg.h_total +
    feed_air_fluidisation.massflow*feed_air_fluidisation.h_total +
    feed_air_primary.massflow*feed_air_primary.h_total +
    feed_air_secondary.massflow*feed_air_secondary.h_total +
    feed_gas.massflow*feed_gas.h_total + feed_org.massflow*feed_org
    .h_total;
```

The standard unit for massflow in IPSEpro is kg/h and kJ/kg_gas for the total enthalpy for gaseous streams, both are defined within the gas connection source code. Therefore, for the energy balance (Code 4.96) Q_{loss} (kW) is multiplied by the factor of 3600.

Code 4.97: Thermal power of the combustion reactor based on lower heating values

```
# Case 2: Char referenced in drain
ifl ref(drain.Char) then
fP_therm_2: P_therm*3600.0      = feed_bed.massflow*feed_bed.
    char_content*feed_bed.Char.lhv - drain.nvolflow*(drain.
    char_content/1000)*drain.Char.lhv + feed_fg.nvolflow*feed_fg.
    Gas.LHV*1000 + feed_fg.nvolflow*(feed_fg.char_content/1000)*
    feed_fg.Char.lhv + feed_air_fluidisation.nvolflow*
    feed_air_fluidisation.Gas.LHV*1000 + feed_air_primary.nvolflow*
    feed_air_primary.Gas.LHV*1000 + feed_air_secondary.nvolflow*
    feed_air_secondary.Gas.LHV*1000 + feed_gas.nvolflow*(feed_gas.
    Gas.LHV*1000 + (feed_gas.tar_content/1000)*feed_gas.Tar.lhv) +
    feed_org.massflow*feed_org.lhv;
endifl
```

The thermal power oft the combustion reactor is calculated by the lower heating value (*LHV*). In the unit's source code two cases are separated. Either there is char in the gaseous drain stream or not. Code 4.97 shows the case when char is referenced in the flue gas stream. For the other case the equation of Code 4.97 is reduced by the char content of the drain gas stream.
In the gaseous drain stream the share of organic char is defined in g/Nm^3 and the *lhv* for organic components is defined in kJ/kg. In contrast to the *LHV* value for gas, it is set -by default- in MJ/Nm^3. In order to meet the unit identity for the balance, the organic part is divided by the factor of 1000 and the gaseous one multiplied by the same factor. *P_therm* is multiplied by the factor of 3600 to convert into the unit of kJ/h.

Code 4.98: Thermal power loss of the combustion reactor based on lower heating values

```
fQ_loss: Q_loss = P_therm*q_loss_rel/100;
```

The relative heat loss is calculated or set due to Code 4.98. In PSE it is presented in % and therefore divided by the factor of 100 (Equation 4.98).

Code 4.99: Exergy loss of the combustion reactor based on lower heating values

```
fE_loss: drain.Exergy + E_loss = feed_bed.Exergy + feed_fg.Exergy
    + feed_air_fluidisation.Exergy + feed_air_primary.Exergy +
    feed_air_secondary.Exergy + feed_gas.Exergy + feed_org.Exergy;
```

The Exergy loss is obtained due to a simple balance of Code 4.99.

Code 4.100: Pressure drops for the combustion reactor

```
fdp_gas: feed_gas.p = drain.p + dp_gas;
fdp_org: feed_org.p = drain.p + dp_org;
```

The organic and gas pressure drop is calculated through the Equations of Code 4.100

Code 4.101: Char slips to the drain stream

```
fChar_slip: drain.nvolflow*(drain.char_content/1000.0) = (
    feed_bed.massflow*feed_bed.char_content + feed_fg.nvolflow*(
    feed_fg.char_content/1000.0) ) * (char_slip/100);
```

Char splitting, one part slips to the drain stream and the other part will be combusted. The factor of 1000.0 (Equation of Code 4.97) is initiated as the char content is set in g/Nm^3. Lambda is defined as the actually available amount of oxygen divided by the stoichiometric required amount . Equation 4.8 is relevant for Code 4.102.

$$\lambda = \frac{m_{available}}{m_{required}} \tag{4.8}$$

Code 4.102: Lambda - Char not referenced in drain stream

```
ifl !ref(drain.Char) then
flambda_1: lambda * drain.massflow*(drain.Gas.wO2 - drain.Gas.wCO
    *31.9988/56.0208) = (lambda - 1.0)*(feed_fg.massflow*feed_fg.
    Gas.wO2 + feed_air_fluidisation.massflow*feed_air_fluidisation.
    Gas.wO2 + feed_air_primary.massflow*feed_air_primary.Gas.wO2 +
    feed_air_secondary.massflow*feed_air_secondary.Gas.wO2 +
    feed_gas.massflow*feed_gas.Gas.wO2);
endifl
```

Code 4.103: Lambda - Char referenced in drain stream

```
ifl ref(drain.Char) then
flambda_2: lambda * (drain.massflow*(drain.Gas.wO2 - drain.Gas.wCO
    *31.9988/56.0208) - drain.nvolflow*(drain.char_content/1000.0)
    * (drain.Char.wC*2*15.9994/12.011 + drain.Char.wH
    *15.9994/(2*1.00794) - drain.Char.wO)) = (lambda - 1.0)*(
    feed_fg.massflow*feed_fg.Gas.wO2 + feed_air_fluidisation.
    massflow*feed_air_fluidisation.Gas.wO2 + feed_air_primary.
    massflow*feed_air_primary.Gas.wO2 + feed_air_secondary.massflow
    *feed_air_secondary.Gas.wO2 + feed_gas.massflow*feed_gas.Gas.
    wO2);
endifl
```

The equations of Code 4.102 and 4.103 are derived from Equation 4.8 via replacing $m_{required}$ by the term for excess oxygen ($m_{available} - m_{required} = m_{excess}$). In case of Code 4.103, it is assumed that the only char elements participating in the combustion procedure are C and H. Therefore S, Cl and N are not considered for the calculation of lambda.

Code 4.104: Char referenced in drain stream

```
fw_riser: w_riser*A_riser = drain.opvolflow/3600.0;
```

The superficial gas velocity in the riser is created through the equation of Code 4.104. The effective sectional area of the riser has to be defined first, or vice versa.

Code 4.105: Char referenced in drain stream

```
fGs: Gs*3600*A_riser = feed_bed.massflow;
```

The specific solid flow rate (Gs) through the riser is calculated by Code 4.105.

Code 4.106: Dust balances

```
# Case 1: Dust not referenced in fuel gas
ifl !ref(feed_gas.Dust) then
fAsh_1: drain.nvolflow*(drain.dust_content/1000.0)*drain.Dust.wAsh
    = feed_bed.massflow*feed_bed.Solid.wAsh + feed_fg.nvolflow*(
    feed_fg.dust_content/1000.0)*feed_fg.Dust.wAsh
        + feed_org.massflow*feed_org.ash_content*feed_org.Ash.wAsh
            ;
fK2O_1: drain.nvolflow*(drain.dust_content/1000.0)*drain.Dust.wK2O
    = feed_bed.massflow*feed_bed.Solid.wK2O
        + feed_fg.nvolflow*(feed_fg.dust_content/1000.0)*feed_fg.
            Dust.wK2O + feed_org.massflow*feed_org.ash_content*
            feed_org.Ash.wK2O;
                        .
                        .
fCaOH2_1:       drain.nvolflow*(drain.dust_content/1000.0)*drain.
    Dust.wCaOH2 = feed_bed.massflow*feed_bed.Solid.wCaOH2
        + feed_fg.nvolflow*(feed_fg.dust_content/1000.0)*feed_fg.
            Dust.wCaOH2 + feed_org.massflow*feed_org.ash_content*
            feed_org.Ash.wCaOH2;
endifl

# Case 2: Dust referenced in fuel gas
ifl ref(feed_gas.Dust) then
fAsh_2: drain.nvolflow*(drain.dust_content/1000.0)*drain.Dust.wAsh
    = feed_bed.massflow*feed_bed.Solid.wAsh + feed_fg.nvolflow*(
    feed_fg.dust_content/1000.0)*feed_fg.Dust.wAsh
        + feed_gas.nvolflow*(feed_gas.dust_content/1000.0)*
            feed_gas.Dust.wAsh + feed_org.massflow*feed_org.
            ash_content*feed_org.Ash.wAsh;
                        .
                        .
fCaOH2_2:       drain.nvolflow*(drain.dust_content/1000.0)*drain.
    Dust.wCaOH2 = feed_bed.massflow*feed_bed.Solid.wCaOH2
        + feed_gas.nvolflow*(feed_gas.dust_content/1000.0)*
            feed_gas.Dust.wCaOH2 + feed_fg.nvolflow*(feed_fg.
            dust_content/1000.0)*feed_fg.Dust.wCaOH2 + feed_org.
            massflow*feed_org.ash_content*feed_org.Ash.wCaOH2;
endifl
```

Inorganic solids are treated as chemically invariant. A balance for the inorganic species defines the drain gas composition of the inorganic components. Two different scenarios are considered. For the first one dust is not referenced in the fuel gas (stream *feed_gas*), for the second one dust is assumed as a part of the fuel gas, see Code 4.106.
Like for any other gaseous streams, also the gaseous drain stream composition of this unit is defined by 18 different species. Therefore, 18 equations are required to solve this system of equations. One of them is already provided by Code 4.45. It is included in the gas global itself and therefore applied by default for every gaseous stream. The sum of all mole shares equals one. The remaining equations are listed in the following part.

Code 4.107: No gaseous organic carbon in flue gas

```
frC2H4: drain.Gas.yC2H4 = 0.0;
frC2H6: drain.Gas.yC2H6 = 0.0;
frC3H8: drain.Gas.yC3H8 = 0.0;
frCH4:  drain.Gas.yCH4 = 0.0;
```

Code 4.108: No hydrogen, hydrogen sulfide, hydrogen cyanide, nitrous oxide, or ammonia in flue gas

```
frH2:   drain.Gas.yH2 = 0.0;
frH2S:  drain.Gas.yH2S = 0.0;
frHCN:  drain.Gas.yHCN = 0.0;
frN2O:  drain.Gas.yN2O = 0.0;
frNH3:  drain.Gas.yNH3 = 0.0;
```

Assumptions are made that no gaseous organic carbon, hydrogen, hydrogen sulfide, hydrogen cyanide, nitrous oxide, or ammonia are present in the flue gas (Code 4.107/4.108).
Other species shares are calculated due to an elementary balance. For the elementary balance four different cases are considered, depending on the presence of char in various streams, Code 4.109.

Code 4.109: Elementary mass balances

```
fAr:    drain.massflow*drain.Gas.wAr = feed_fg.massflow*feed_fg.
    Gas.wAr + feed_air_fluidisation.massflow*feed_air_fluidisation.
    Gas.wAr + feed_air_primary.massflow*feed_air_primary.Gas.wAr +
    feed_air_secondary.massflow*feed_air_secondary.Gas.wAr +
    feed_gas.massflow*feed_gas.Gas.wAr;

# Case 1: Char not referenced in drain
ifl !ref(drain.Char) then
fC_1:   ....
                        .
                        .
endifl
# Case 2: Char referenced in drain
ifl ref(drain.Char) then
fC_2:   ....
                        .
                        .
endifl
```

```
# Case 3: Char not referenced in drain and referenced in fuel gas
ifl !ref(drain.Char) && ref(feed_gas.Char) then
fC_3:    ....
                          .
                          .
endifl
# Case 4: Char referenced in drain and in fuel gas
ifl ref(drain.Char) && ref(feed_gas.Char) then
fC_4:    ....
                          .
                          .
endifl
```

Code 4.110: NOx formation in the riser

```
ifl NOx_formation == NONE then
fr1N: drain.nvolflow*drain.Gas.yN2 = feed_fg.nvolflow*feed_fg.Gas.
    yN2 + feed_air_fluidisation.nvolflow*feed_air_fluidisation.Gas.
    yN2 + feed_air_primary.nvolflow*feed_air_primary.Gas.yN2 +
    feed_air_secondary.nvolflow*feed_air_secondary.Gas.yN2 +
    feed_gas.nvolflow*feed_gas.Gas.yN2;
endifl
ifl NOx_formation == DEF_GSSG then
fr2N: drain.Gas.yNO      = 1.5e-4;
endifl
```

Through a switch function it can be selected if either NO_x is formed or not. In case of no NO_x formation, no thermal NO is formed so that N_2 passes without reacting. When choosing the option of NO_x production, a fixed value for thermal NO_x according to total NO_x measurement from Guessing -by T. Proell for January 1st 2005-, is set.
The energy and mass balance are taken from the simulation of Section 3.1, for the illustration of Figure 4.16. In both cases there is no relevant *Difference* between input and output.

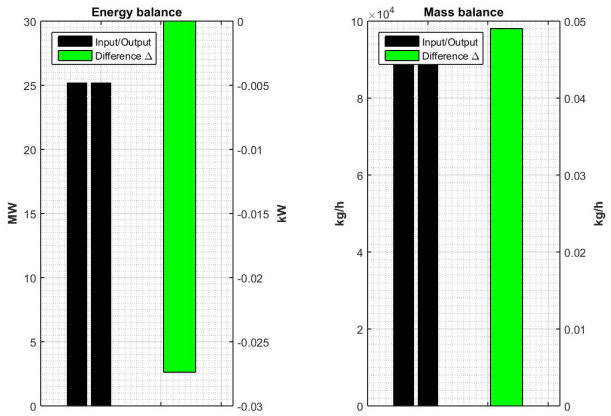


Figure 4.16: Energy and mass balance for the combustion chamber

Combustion-afterburner u_comb_g_

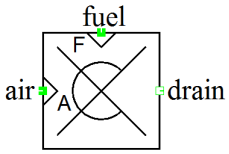


Figure 4.17: Combustion-afterburner library icon

This model of a gas combustion chamber is optionally allowing dust, char, and tar in gaseous fuel. No conversions of inorganic solids occurs for the following Code description, no dust/char/tar are present in the combustion air, no char or tar are assumed in the flue gas stream. If dust is referenced in the *fuel* stream, it has to be referenced in the *drain* stream as well. Incomplete combustion is expressed by CO in drain. Therefore, the variable *CO_slip* is defined. At air ratios smaller one, the CO_slip determines the O_2-content in the *drain* stream. Air is fed by the inlet *air*. Additional information on the afterburner unit can be found in [76][5.1.6]. The implementation of the variable *char_slip* is described in detail by J. Kotik [60][3.2.2.2].

Switch	Description
NOx_formation	Model allows description of thermal NOx formation according to the switch position
Variables	**Description**
dp_fuel	Pressure drop from fuel gas to drain [bar]
dp_fuel_rel	Relative pressure drop for fuel based on feed pressure [%]
dp_air	Pressure drop from air to drain [bar]
dp_air_rel	Relative pressure drop for air based on feed pressure [%]
lambda	Air ratio of the combustion chamber based on the amount of oxygen in air stream needed for stoichiometric combustion of all substances entering the combustion chamber (e.g. also combustables entering with the air stream) [-]
CO_slip	Molar ratio between CO and CO_2 in flue gas. This value must be determined by the system (set or exiting CO concentration set) if lambda ≥ 1.0. If lambda < 1.0 the CO_slip has no influence on the system and must be set to a dummy value. [%CO_2]
char_slip	Ratio between uncombusted char in drain gas and incoming char in fuel [%]
P_therm	Thermal power of the combustion chamber based on lower heating values of all combustibles entering the reactor [kW]
q_loss_rel	Relative heat loss of the combustion reactor in % of thermal power [%]
Q_loss	Heat loss of the combustion reactor [kW]
E_loss	Exergy loss caused by the combustion process [kW]

Table 4.20: Combustion-afterburner item description

Code 4.111: Energy balance of the combustion chamber

```
fEnergy:        drain.massflow*drain.h_total + Q_loss*3600.0 = air
    .massflow*air.h_total + fuel.massflow*fuel.h_total;
```

Code 4.112: Thermal power of the combustion reactor

```
ifl !ref(fuel.Char) && !ref(fuel.Tar) then
fP_therm_1:     P_therm*3600.0  = air.nvolflow*air.Gas.LHV*1000.0
    + fuel.nvolflow*fuel.Gas.LHV*1000.0;
endifl
                                .
                                .
ifl ref(fuel.Char) && ref(drain.Char) && ref(fuel.Tar) then
fP_therm_6:     P_therm*3600.0  = air.nvolflow*air.Gas.LHV*1000.0
    + fuel.nvolflow*(fuel.Gas.LHV*1000.0 + (fuel.char_content
    /1000.0) *fuel.Char.lhv + (fuel.tar_content/1000.0)*fuel.Tar.
    lhv) - drain.nvolflow*(drain.char_content/1000)*drain.Char.lhv;
endifl
```

Different scenarios considering the presence of char and tar are considered for the generation of the thermal power in Code 4.112.

Code 4.113: Heat loss of the combustion reactor

```
fQ_loss:        Q_loss = P_therm*(q_loss_rel/100.0);
```

Code 4.114: Exergy loss due to irreversible combustion

```
fE_loss:        drain.Exergy + E_loss = air.Exergy + fuel.Exergy;
```

Code 4.115: Pressure drops

```
fdp_air:        air.p = drain.p + dp_air;
fdp_air_rel:    0.01*dp_air_rel = dp_air/air.p;
fdp_fuel:       fuel.p = drain.p + dp_fuel;
fdp_fuel_rel:   0.01*dp_fuel_rel = dp_fuel/fuel.p;
```

Variables of Table 4.20 are defined by Code 4.113 to 4.115.

Code 4.116: Elementary mass balances

```
fAr_1:  drain.massflow*drain.Gas.wAr = air.massflow*air.Gas.wAr +
    fuel.massflow*fuel.Gas.wAr;
                                .
                                .
fCl_1:  drain.massflow*35.4527*drain.Gas.wHCl/36.46064
                = air.massflow*35.4527*air.Gas.wHCl/36.46064
                + fuel.massflow*35.4527*fuel.Gas.wHCl/36.46064;
endifl
                                .
                                .
```

To generate the *drain* gas composition, the elementary mass balances have to be fulfilled. Equal to the variable *P_therm*, various cases are considered in dependency if dust, char and tar are present or not.

Code 4.117: Char splitting - one part slips to the drain and the other part will be combusted

```
fChar_slip:      if fuel.char_content > 0.000001 then     drain.
    nvolflow*(drain.char_content/1000.0) = fuel.nvolflow*(fuel.
    char_content/1000.0) *(char_slip/100);
else     char_slip = 0.0;
```

Code 4.118: Inorganic solids (dust) section

```
ifl !ref(air.Dust) && ref(fuel.Dust) && ref(drain.Dust)
        && ref(fuel.Dust) == ref(drain.Dust)
then
        fdust_cont:      fuel.nvolflow*fuel.dust_content = drain.
            nvolflow*drain.dust_content;
endifl
                         .
                         .
```

As before, also for the generation of the drain dust composition different scenarios for the presents of dust, char and tar are considered.

Code 4.119: CO slip

```
ifl !ref(drain.Char)     then fLambda_1: lambda * drain.massflow*(
    drain.Gas.wO2 - drain.Gas.wCO*31.9988/56.0208) = (air.massflow*
    air.Gas.wO2 + fuel.massflow*fuel.Gas.wO2) * (lambda - 1.0);
elsel fLambda_2: lambda * (drain.massflow*(drain.Gas.wO2 - drain.
    Gas.wCO*31.9988/56.0208) - drain.nvolflow*(drain.char_content
    /1000.0) * (drain.Char.wC*2*15.9994/12.011 + drain.Char.wH
    *15.9994/(2*1.00794) - drain.Char.wO)) = (air.massflow*air.Gas.
    wO2 + fuel.massflow*fuel.Gas.wO2) * (lambda - 1.0);
endifl
```

Code 4.120: No gaseous organic carbon in flue gas

```
frC2H4: drain.Gas.yC2H4 = 0.0;
frC2H6: drain.Gas.yC2H6 = 0.0;
frC3H8: drain.Gas.yC3H8 = 0.0;
frCH4:  drain.Gas.yCH4 = 0.0;
```

Code 4.121: Free oxygen

```
frCO:    if lambda >= 1.0          then drain.Gas.yCO = (CO_slip
    /100.0)*drain.Gas.yCO2;
else drain.Gas.yO2 = 0.5*(CO_slip/100.0)*drain.Gas.yCO2;
```

Code 4.122: No hydrogen, hydrogen sulfide, hydrogen cyanide, nitrous oxide, or ammonia in flue gas

```
frH2:    drain.Gas.yH2 = 0.0;
frH2S:   drain.Gas.yH2S = 0.0;
frHCN:   drain.Gas.yHCN = 0.0;
frN2O:   drain.Gas.yN2O = 0.0;
frNH3:   drain.Gas.yNH3 = 0.0;
```

Different assumptions for the *drain* gas composition are specified from Code 4.119 to 4.122.

Code 4.123: No hydrogen, hydrogen sulfide, hydrogen cyanide, nitrous oxide, or ammonia in flue gas

```
if1 NOx_formation == None then
        frN2:    drain.nvolflow*drain.Gas.yN2 = air.nvolflow*air.
            Gas.yN2 + fuel.nvolflow*fuel.Gas.yN2;
endif1
```

If the switch option *None* is selected, no thermal NO is formed and N_2 passes without reacting.

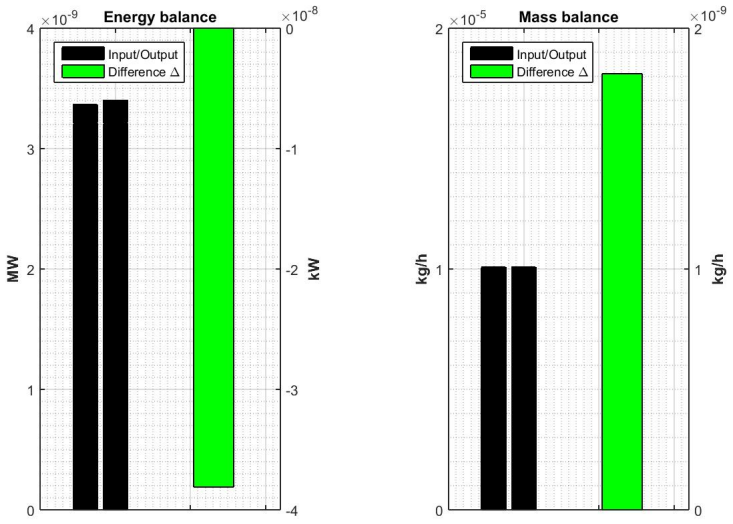


Figure 4.18: Energy and mass balance for the afterburner chamber

Compressor gas u_compr_g_

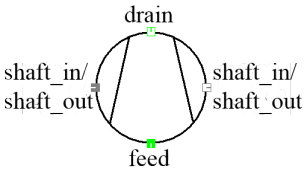


Figure 4.19: Compressor library icon

Besides gaseous streams, a compressor model is also available for water respectively steam. Exemplary the gas model is presented in this subsection. Equally to any other shaft connection, the terminal is implemented in rotary mode. This means that the inlet is always connected from the left side even if the icon is rotated. Therefore, each shaft connection is defined as an in- and out-let stream *shaft_in/shaft_out*. The gas inlet is called *feed* and the outlet *drain*. Dust, char, and tar can be optionally referenced. If referenced, the same global objects have to be used in the *feed* and *drain* stream. Additional information can be found in [76][5.1.2].

Variables	Description
dp	Pressure difference between drain and feed [bar]
press_ratio	Pressure ratio of the compressor [-]
eta_s	Isentropic efficiency [%]
eta_m	Mechanical efficiency [%]
P_rotor_disc	Power at the rotor disc, which will be absorbed by the fluid [kW]
E_loss	Exergy loss [kW]
ts	Hypothetical temperature after isentropic compression [°C]

Table 4.21: Compressor item description

Code 4.124: Increase of pressure

```
fdp:    drain.p = feed.p + dp;
```

Code 4.125: Mass balance for gas

```
fmass:  feed.massflow = drain.massflow;
```

Code 4.126: Mass balance for dust, char and tar

```
ifl ref(feed.Dust) || ref(drain.Dust) then fmass_dust:  feed.
    dust_content = drain.dust_content; endifl
ifl ref(feed.Char) || ref(drain.Char) then fmass_char:  feed.
    char_content = drain.char_content; endifl
ifl ref(feed.Tar) || ref(drain.Tar) then fmass_tar:     feed.
    tar_content = drain.tar_content;   endifl
```

The mass balance for dust, char and tar is considered in Code 4.126.

Code 4.127: Hypothetical temperature after isentropic compression implicitely calculated from enthalpy after hypothetical isentropic compression

```
fts:    feed.h + (drain.h - feed.h)*eta_s/100 = drain.Gas.gfht(ts)
    ;
```

Code 4.128: Irreversible compression

```
fcompr: feed.s = drain.Gas.gfspt(drain.p, ts);
```

Some of the variables of Table 4.21 are defined by Code 4.127, 4.128, 4.130 and 4.131.

Code 4.129: Elementary balances

```
if1 ref(shaft_in) && ref(shaft_out) then
        f1h:    (drain.h_total - feed.h_total)*feed.massflow/(
            eta_m/100.0) = (shaft_in.power - shaft_out.power)
            *3600.0;
        f1_E_loss:      E_loss = feed.Exergy - drain.Exergy +
            shaft_in.power - shaft_out.power;
endif1
if1 ref(shaft_in) && !ref(shaft_out) then
        f2h:    (drain.h_total - feed.h_total)*feed.massflow/(
            eta_m/100.0) = shaft_in.power*3600.0;
        f2_E_loss:      E_loss = feed.Exergy - drain.Exergy +
            shaft_in.power;
endif1
if1 !ref(shaft_in) && ref(shaft_out) then
        f3h:    (drain.h_total - feed.h_total) * feed.massflow/(
            eta_m/100.0) = - shaft_out.power*3600.0;
        f3_E_loss:      E_loss = feed.Exergy - drain.Exergy -
            shaft_out.power;
endif1
```

The energy balance is defined by Code 4.129. Different shaft connection scenarios are considered. Both sides are connected for the first case, for the second one just the left side is connected, in the last case it is just connected from the right side.

Code 4.130: Power of rotor disc

```
f_p_rotor_disc:         P_rotor_disc = (drain.h_total - feed.
    h_total)*feed.massflow/3600;
```

Code 4.131: Pressure ratio

```
fpress_ratio:   feed.p*press_ratio = drain.p;
endif1
```

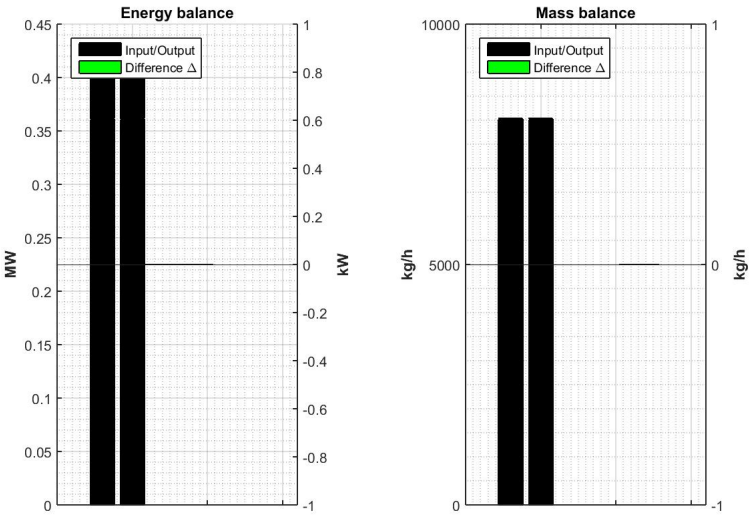


Figure 4.20: Energy and mass balances for the compressor

PEM - Electrolyser u_electr_pem_

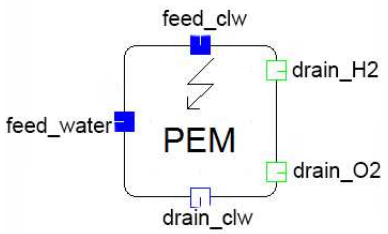


Figure 4.21: PEM library icon

Figure 4.21 shows the symbol of the proton exchange membrane (PEM) electrolyser model in the BG-Lib library file. The inlet *feed_water* provides deionized (DI) water to the electrolysis process, whereas the inlet *feed_clw* is installed for cooling water supply. The heated cooling water leaves the electrolyser through exit *drain_clw*. The drain connection *drain_H2* transports hydrogen and the exit *drain_O2* oxygen. It is assumed that both products are 100% pure. Mainly because of its capability to handle changes in electricity supply very well, the PEM-electrolyser has been chosen for recent Power-to-Gas (PtG) applications.
The PEM-electrolyser model is designed to be able to implement data according to technical specifications. Table 4.22 shows the list of items to be minded when using the PSE-interface of the PEM-electrolyser model.

Variables	Description
dp_clw	Pressure loss of coolant through PEM electrolyser [bar]
net_prod_rate	Consumed power to produce 1 Nm^3 of H_2 gas [kWh/Nm^3]
Conv_Efficiency	Conversion efficiency of the electrolyser [%]
P_el_in	Electrical power consumed [kW]
Q_trans	Heat transferred from electrolyser stack to cooling medium [kW]
Vn_H2	Standard volume flow of H_2 [Nm^3/h]
T_H2	Temperature of H_2 product stream [°C]
p_H2	Pressure of H_2 product stream [bar]
T_O2	Temperature of O_2 product stream [°C]
p_O2	Pressure of O_2 product stream [bar]
E_Loss	Exergy loss [kW]
Global	**Description**
amb	Geographical and meteorological conditions

Table 4.22: PEM electrolyser item description

The total power consumption for hydrogen and oxygen generation (*P_el_in*) is calculated as consumed power per volume of H_2-gas times the standard volume flow *Vn_H2* (Code 4.132). *Vn_H2* is usually provided by the respective product specification of the individual manufacturer. The conversion efficiency of Code 4.133 is based on Smolinka et. al. [96] but slightly modified. The basic definition is just considering the chemical energy of hydrogen as a product. In order to meet the energy balance, for this model also the chemical energy of oxygen as well as the sensible heat of hydrogen and oxygen are considered for the efficiency calculation. The simulation results between the basic definition according to Smolinka et. al. [96] and the modified one (Code 4.133) are marginal so that efficiencies from literature can be used for the model.

Code 4.132: Consumed power per volume of mass H_2 gas produced

```
fsupply_electrical_power:        P_el_in = Vn_H2 * net_prod_rate;
```

Code 4.133: Conversion Efficiency

```
fefficiency:     Conv_Efficiency * P_el_in *3600  = ((drain_H2.
    lhv_total + drain_H2.q_298_total) * drain_H2.massflow + (
    drain_O2.lhv_total + drain_O2.q_298_total) * drain_O2.massflow)
     * 100;
```

Code 4.134: Process water demand

```
fpr_water_demand_massflow: feed_water.massflow = drain_H2.massflow
     + drain_O2.massflow;
```

Code 4.135: Energy balance

```
f_energy_balance: Q_trans * 3600 = P_el_in *3600 -(drain_H2.
    massflow * drain_H2.h_total) -(drain_O2.h_total * drain_O2.
    massflow) + (feed_water.massflow * feed_water.h_total);
```

The power loss to the environment can be neglected (personal information of Markus Koppe, JKU Linz) for the electrolyser model. Therefore, *Q_loss* is not considered for the energy balance (Code 4.135).

Code 4.136: Energy balance cooling media

```
ftemp_cooling:   drain_clw.massflow/3600*drain_clw.h_total =
    Q_trans + feed_clw.massflow/3600*feed_clw.h_total;
```

Q_trans describes the energy transferred to the cooling media (Code 4.136).

Code 4.137: Norm volume flows

```
fnorm_volume_flow_H2:    drain_H2.nvolflow = Vn_H2;
fnorm_volume_flow_O2:    drain_O2.nvolflow = Vn_H2 * 0.5;
```

The produced amount of hydrogen and oxygen is defined by Code 4.137. It is related to the chemical reaction for water electrolysis which appears at the cathode side of the PEM-electrolyser due to recombination of electrons and protons [5].

Code 4.138: Pressure and temperature of product streams

```
fp_H2:          drain_H2.p = p_H2;
fT_H2:          drain_H2.t = T_H2;
fp_O2:          drain_O2.p = p_O2;
fT_O2:          drain_O2.t = T_O2;
```

Code 4.139: Pressure drop cooling media

```
fp_drain_cooling_water:          drain_clw.p = feed_clw.p - dp_clw;
```

Code 4.140: Mass balance cooling media

```
fmb_cooling2:                    feed_clw.massflow = drain_clw.
    massflow;
```

Code 4.141: Composition H_2-stream

```
fmassgcAr_H2:    drain_H2.Gas.wAr          = 0.0;
fmassgcC2H4_H2:  drain_H2.Gas.wC2H4        = 0.0;
                          .
                          .
fmassgcO2_H2:    drain_H2.Gas.wO2          = 0.0;
fmassgcSO2_H2:   drain_H2.Gas.wSO2         = 0.0;
```

Code 4.142: Composition O_2-stream

```
fmassgcAr_O2:    drain_O2.Gas.yAr          = 0.0;
fmassgcC2H4_O2:  drain_O2.Gas.yC2H4        = 0.0;
                          .
fmassgcH2_O2:    drain_O2.Gas.yH2          = 0.0;
                          .
fmassgcNO_O2:    drain_O2.Gas.yNO          = 0.0;
fmassgcSO2_O2:   drain_O2.Gas.ySO2         = 0.0;
```

The compositions of the H_2 and O_2 drain streams are defined within the MDK-electrolyser source code (Code 4.141 and 4.142). For both product streams all species, except the desired one, are set to zero so that an product purity of 100% is calculated through Code 4.45.

Code 4.143: Exergy Loss

```
f_Exergy:        P_el_in + feed_water.Exergy + feed_clw.Exergy =
    drain_H2.Exergy + drain_O2.Exergy + drain_clw.Exergy + E_Loss;
```

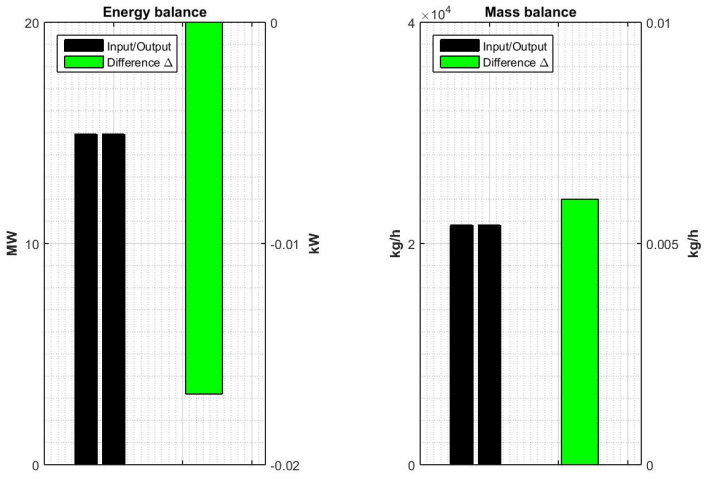


Figure 4.22: Energy and mass balance for the PEM-electrolyser

Te PEM-model is applied for the simulation of Section 3.1. Therefore, the energy and mass balance of Figure 4.22 are taken from this simulation. In both cases there is no significant *Difference* between input and output.

Methanation reactor unit u_react_g_

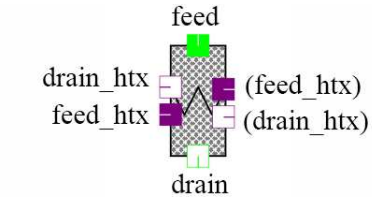


Figure 4.23: Reactor library icon

The methanation reactor model is almost equal to the already existing steam reforming reactor model, created by T. Pröll. Two main adoptions have been made. For each type of cooling media (gaseous, organic, water) a specific reactor model has been created. The second change was the integration of the conversion rates for CO and CO_2.
The gaseous phase is fed via the entrance *feed* and leaves through the exit *drain*. The cooling agent enters the reactor model at the terminal *feed_htx* and leaves at *drain_htx*. It

can be optionally connected to the reactor from both sides. Therefore, in Figure 4.23 there is also *feed_htx* and *drain_htx* within parenthesis.

Variables	Description
dp	Pressure drop from fuel gas to drain [bar]
dt	Temperature decrease from feed to drain: dt = feed.t - drain.t [°C]
dp_htx	Pressure drop of cooling media [bar]
H_sens_in	Total sensible heat of entering stream [kW]
Q_htx	Heating(+)/cooling(-) power [kW]
q_loss_rel	Relative heat loss of the methanation reactor in % of sensible heat input [%]
Q_loss	Heat loss of the combustion reactor [kW]
E_loss	Enthalpy of reaction (endothermic reactions count positively, exothermic reactions negatively) [kW]
H_react	Heat loss of the combustion reactor [kW]
X_C2H4	Conversion rate of C_2H_4 [%]
X_C2H6	Conversion rate of C_2H_6 [%]
X_C3H8	Conversion rate of C_3H_8 [%]
X_CO	Conversion rate of CO [%]
X_CO2	Conversion rate of CO_2 [%]
Function	**Description**
Kp_CO_H2O_CO2_H2	Function returns equilibrium constant for partial pressures of the water gas shift reaction

Table 4.23: Reactor item description

Code 4.144: Energy balance of the reactor

```
# Case 1: No additional heating/cooling referenced
if1 !ref(feed_htx) || !ref(drain_htx) then
        f1Energy:       drain.massflow*drain.h_total + Q_loss
            *3600.0 = feed.massflow*feed.h_total;
# Case 2: Additional heating/cooling referenced
else1
        f2Energy:       drain.massflow*drain.h_total + Q_loss
            *3600.0 + drain_htx.massflow*drain_htx.h_total = feed.
            massflow*feed.h_total + feed_htx.massflow*feed_htx.
            h_total;
endif1
```

Code 4.145: Sensible heat entering the reactor

```
# Case 1: No additional heating/cooling referenced
if1 !ref(feed_htx) || !ref(drain_htx) then
        f1H_sens_in: H_sens_in*3.6e+3 = feed.massflow*feed.h + (
            feed.nvolflow*0.001)*(feed.dust_content*feed.h_dust +
            feed.char_content*feed.h_char + feed.tar_content*feed.
            h_tar);
```

```
# Case 2: Additional heating/cooling referenced
elsel
        f2H_sens_in: H_sens_in*3.6e+3 = feed.massflow*feed.h + (
            feed.nvolflow*0.001)*(feed.dust_content*feed.h_dust +
            feed.char_content*feed.h_char + feed.tar_content*feed.
            h_tar) + feed_htx.massflow*feed_htx.h_waf;
endifl
```

Code 4.146: Reaction enthalpy

```
fH_react:       H_react = H_sens_in - (drain.massflow*drain.h + (
    drain.nvolflow*0.001)*(drain.dust_content*drain.h_dust + drain.
    char_content*drain.h_char + drain.tar_content*drain.h_tar))/3.6
    e+3 - Q_loss;
```

Code 4.147: Power of heating/cooling

```
ifl ref(feed_htx) && ref(drain_htx) then
        f1Q_htx:        drain_htx.massflow*drain_htx.h_waf =
            feed_htx.massflow*feed_htx.h_waf - 3600.0*Q_htx;
elsel
        f2Q_htx:        Q_htx = 0.0;
endifl
```

Code 4.148: Heat loss of the methanation reactor

```
fQ_loss:        Q_loss = H_sens_in*(q_loss_rel/100.0);
```

Code 4.149: Exergy loss due to irreversible combustion

```
# Case 1: No additional heating/cooling referenced
ifl !ref(feed_htx) || !ref(drain_htx) then
        f1E_loss:       drain.Exergy + E_loss = feed.Exergy;
# Case 2: Additional heating/cooling referenced
elsel
        f2E_loss: drain.Exergy + E_loss + drain_htx.Exergy = feed.
            Exergy + feed_htx.Exergy;
endifl
```

Code 4.150: Pressure drop gas stream

```
fdp:    feed.p = drain.p + dp;
```

Code 4.151: Temperature decrease from feed to drain

```
fdt:    feed.t = drain.t + dt;
```

Code 4.152: Pressure drop for optional heating/cooling

```
ifl ref(feed_htx) && ref(drain_htx) then
        f1dp_htx:       feed_htx.p = drain_htx.p + dp_htx;
elsel
```

```
        f2dp_htx:       dp_htx = 0.0;
endifl
```

Code 4.153: Mass balance for heating/cooling

```
ifl ref(feed_htx) && ref(drain_htx) then
        fmass_htx:      drain_htx.massflow = feed_htx.massflow;
endifl

ifl ref(feed_htx) && ref(drain_htx) && ref(feed_htx.Ash) then
        fash_cont_htx:  drain_htx.ash_content = feed_htx.
            ash_content;
endifl
```

Code 4.154: Elementary mass balances - Neither Char nor Tar in feed

```
ifl !ref(feed.Char) && !ref(feed.Tar) then
fAr_1:   drain.massflow*drain.Gas.wAr = feed.massflow*feed.Gas.wAr;
fC_1:    drain.massflow*12.011*(2*drain.Gas.wC2H4/28.0536 + 2*drain
    .Gas.wC2H6/30.0694 + 3*drain.Gas.wC3H8/44.0962 + drain.Gas.wCH4
    /16.0428 + drain.Gas.wCO/28.0104 + drain.Gas.wCO2/44.0098 +
    drain.Gas.wHCN/27.02568) = feed.massflow*12.011*(2*feed.Gas.
    wC2H4/28.0536 + 2*feed.Gas.wC2H6/30.0694 + 3*feed.Gas.wC3H8
    /44.0962 + feed.Gas.wCH4/16.0428 + feed.Gas.wCO/28.0104 + feed.
    Gas.wCO2/44.0098 + feed.Gas.wHCN/27.02568);
                                    .
                                    .
fS_1:    drain.massflow*32.066*(drain.Gas.wH2S/34.08188 + drain.Gas
    .wSO2/64.0648) = feed.massflow*32.066*(feed.Gas.wH2S/34.08188 +
     feed.Gas.wSO2/64.0648);
fCl_1:   drain.massflow*35.4527*drain.Gas.wHCl/36.46064 = feed.
    massflow*35.4527*feed.Gas.wHCl/36.46064;
endifl
```

The elementary mass balance for Ar, C, H, O, N, S and Cl is described via Code 4.154. The Code describes the scenario of no char and no tar present in the gaseous feed stream. Further scenarios on the presents of char and tar can be seen in the MDK-file.

Code 4.155: Inorganic solids section (Dust)

```
# Case 1: feed and drain use the same Dust object
ifl ref(feed.Dust) && ref(drain.Dust)
        && ref(feed.Dust) == ref(drain.Dust)
then
fdust_cont:     feed.nvolflow*feed.dust_content = drain.nvolflow*
    drain.dust_content;
endifl
# Case 2: feed and drain use different Dust objects
ifl ref(feed.Dust) && ref(drain.Dust)
        && ref(feed.Dust) != ref(drain.Dust)
then
fAsh: feed.nvolflow*feed.dust_content*feed.Dust.wAsh = drain.
    nvolflow*drain.dust_content*drain.Dust.wAsh;
                                    .
```

```
.
fCaSO4: feed.nvolflow*feed.dust_content*feed.Dust.wCaSO4 = drain.
    nvolflow*drain.dust_content*drain.Dust.wCaSO4;
fCaOH2: feed.nvolflow*feed.dust_content*feed.Dust.wCaOH2 = drain.
    nvolflow*drain.dust_content*drain.Dust.wCaOH2;
endifl
```

Normally, the same dust object is used in the feed and drain stream. If not, Case 2 of Code 4.155 is applied. The drain dust content gets determined through the elementary mass balances of all solid species (Table 4.10) together with the equation of Code 4.52 (fmass_sum).

Code 4.156: Pressure drop for optional heating/cooling

```
ifl ref(feed_htx) && ref(drain_htx) then
        f1dp_htx:       feed_htx.p = drain_htx.p + dp_htx;
elsel
        f2dp_htx:       dp_htx = 0.0;
endifl
```

No chemical reactions are considered within the code of the methanation reactor model, to consider them it is necessary to combine the reactor model with the appropriate equilibrium unit for methanation.
The methanation reactor model -as applied in this thesis- considers the water-gas-shift reaction (WGS) Eq.2.22, CO-Methanation Eq.2.20 reaction and the CO_2-Methanation Eq.2.21 reaction. Besides the PEM-electrolyser model, this is the main unit of the simulation of

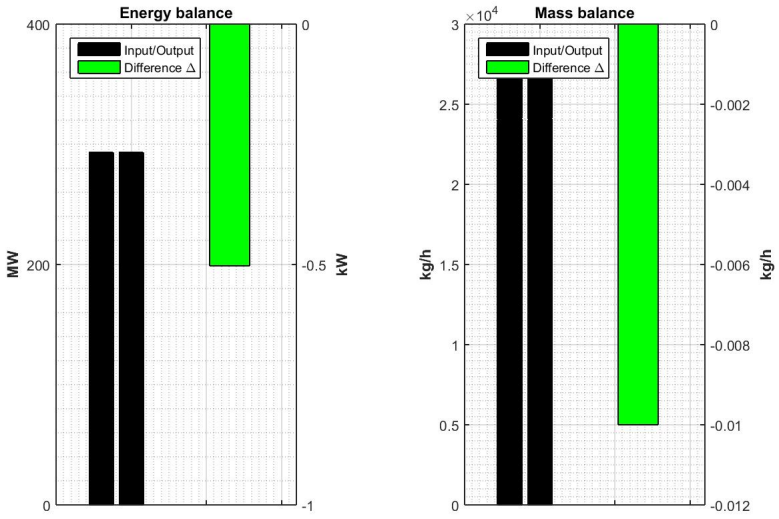


Figure 4.24: Energy and mass balance for the reactor unit

Section 3.1. The energy and mass balance -of Figure 4.24- are results from the simulation of Section 3.1. In both cases there is no significant *Difference* between input and output.

Condenser u_cond_gw_

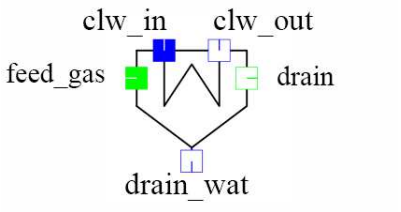


Figure 4.25: Condenser library icon

The already existing condenser model has been extended by the cooling water input *clw_in* and output *clw_out*. The condensing water leaves the model via the exit *drain_wat*. The extension was validated by a mass and energy balance.

Variables	Description
dp_gas	Pressure drop for the gas passing the injector [bar]
dp_wat	Pressure drop for the injected water [bar]
dp_clw	Pressure drop cooling water [bar]
dt_drain	Temperature difference between drain gas and condensate [K]
t_sat_drain	Saturation temperature (dewpoint) in gas drain [K]
dt_sat_drain	Temperature difference from saturation temperature (dewpoint) in gas drain [K]
E_loss	Exergy loss of the injector [kW]
Q_trans	Transferred heat for cooling/condensation [kW]
Function	**Description**
wfp0t	Function returns vapour pressure of H2O [bar] at a given temperature t [°C]

Table 4.24: Condenser item description

Code 4.157: Energy balance

```
fh_total:       drain.massflow*drain.h_total + drain_wat.massflow*
    drain_wat.h_total + 3600.0*Q_trans = feed_gas.massflow*feed_gas
    .h_total;
```

Code 4.158: Energy balance cooling water without influence of construction specific characteristics

```
fclw_h_total:   clw_in.massflow*clw_in.h_total + 3600*Q_trans =
    clw_out.massflow*clw_out.h_total;
```

Code 4.159: Mass balance cooling water

```
fmass_clw:      clw_in.massflow = clw_out.massflow;
```

Code 4.160: Mass balance H_2O

```
fmass_H2O:      drain.massflow*drain.Gas.wH2O + drain_wat.massflow
    = feed_gas.massflow*feed_gas.Gas.wH2O;
```

Code 4.161: Mass balances for the rest of the gas components

```
fmass_Ar:       drain.massflow*drain.Gas.wAr = feed_gas.massflow*
    feed_gas.Gas.wAr;
fmass_C2H4:     drain.massflow*drain.Gas.wC2H4 = feed_gas.massflow
    *feed_gas.Gas.wC2H4;
                                    .
                                    .
                                    .
fmass_O2:       drain.massflow*drain.Gas.wO2 = feed_gas.massflow*
    feed_gas.Gas.wO2;
fmass_SO2:      drain.massflow*drain.Gas.wSO2 = feed_gas.massflow*
    feed_gas.Gas.wSO2;
```

Code 4.162: Mass balance dust

```
ifl ref(drain.Dust) then fmass_dust:    drain.nvolflow*drain.
    dust_content = feed_gas.nvolflow*feed_gas.dust_content;
    endifl
```

Code 4.163: Mass balance char

```
ifl ref(drain.Char) then fmass_char:    drain.nvolflow*drain.
    char_content = feed_gas.nvolflow*feed_gas.char_content;
    endifl
```

Code 4.164: Mass balance tar

```
ifl ref(drain.Tar) then fmass_tar:      drain.nvolflow*drain.
    tar_content = feed_gas.nvolflow*feed_gas.tar_content;
    endifl
```

Code 4.165: Pressure drops

```
fdp_gas:        drain.p = feed_gas.p - dp_gas;
fdp_wat:        drain_wat.p = feed_gas.p - dp_wat;
```

Code 4.166: Temperature difference between drain streams

```
fdt_drain:      drain_wat.t + dt_drain = drain.t;
```

Code 4.167: Saturation temperature (dewpoint) in gas drain

```
ft_sat:         drain.p * drain.Gas.yH2O = wfp0t(t_sat_drain);
```

Code 4.168: Difference between actual temperature and saturation temperature

```
fdt_sat:        dt_sat_drain = drain.t - t_sat_drain;
```

Code 4.169: Exergy loss

```
fExergy: drain.Exergy + drain_wat.Exergy + E_loss = feed_gas.
    Exergy;
```

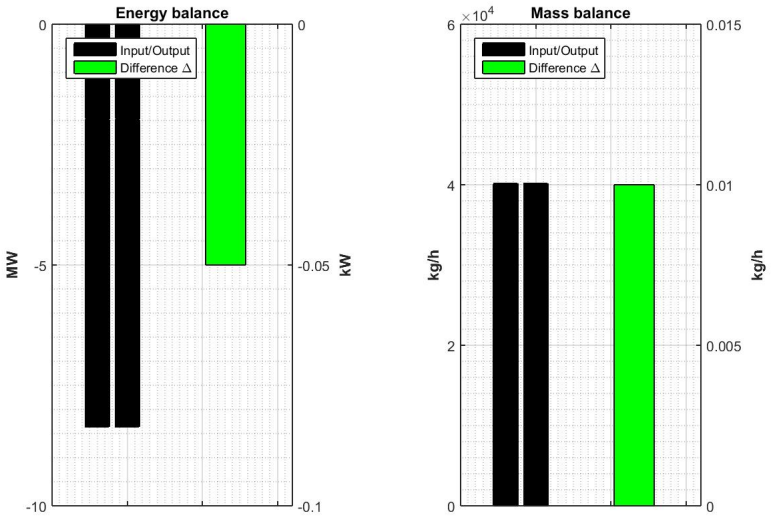


Figure 4.26: Energy and mass balance for the condenser

The energy and mass balance of Figure 4.26 are taken from Section 3.1. For both -energy and mass balance- there is no significant *Difference* between input and output.

Membrane u_membr_g_

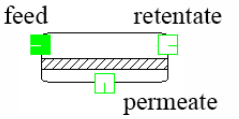


Figure 4.27: Membrane library icon

Figure 4.27 shows the membrane unit's icon. It is fed with gas by the entrance *feed*. The retentate-stream leaves by the exit *retentate* and the permeate-stream through the exit *permeate*. The already existing membrane model was based on the permeability of the various molecules. Specific data of the considered species i (Eq. 4.9) is provided by the working group of Dr. Harasek - Technical University of Vienna. To avoid the definition of the membrane thickness, it is recommended to use the permeance coefficient P_i instead of the permeability coefficient, to describe the transportation process. According to [66] (page 416) the permeability is the ratio of permeance and membrane-thickness.

$$\dot{V}_{n,permeate} \cdot y_{i,permeate} = P_i \cdot A_m \cdot (p_{feed} \cdot y_{i,feed} - p_{permeate} \cdot y_{i,permeate}) \qquad (4.9)$$

Variables	Description
Am	Area of membrane [m^2]
dp_permeate	Pressure drop between feed and permeate [bar]
dp_retentate	Pressure drop between feed and retentate [bar]
Pi	Permeance of the membrane for component i [m/s·bar]
dt_permeat	Temperature difference between feed and permeate stream [K]
dt_retentate	Temperature difference between feed and retentate stream [K]
q_loss_rel	Loss of heat relative to sensible gas heat in feed stream [%]
Q_loss	Heat loss [kW]
E_loss	Exergy loss [kW]

Table 4.25: Membrane model item description

Code 4.170: Mass balances

```
fmass:  feed.massflow = permeat.massflow + retentate.massflow;

f1wAr:  feed.massflow * feed.Gas.wAr = permeat.massflow * permeat.
    Gas.wAr + retentate.massflow * retentate.Gas.wAr;
f1wC2H4: feed.massflow * feed.Gas.wC2H4 = permeat.massflow *
    permeat.Gas.wC2H4 + retentate.massflow * retentate.Gas.wC2H4;
                                    .
                                    .
                                    .
f1wSO2: feed.massflow * feed.Gas.wSO2 = permeat.massflow * permeat
    .Gas.wSO2 + retentate.massflow * retentate.Gas.wSO2;
```

The individual mass balances of all the gaseous species are defined like the few ones presented in Code 4.170.

Code 4.171: Permeances

```
f2wAr:  (permeat.nvolflow/3600) * permeat.Gas.yAr = PAr * Am * (((
    feed.p * feed.Gas.yAr) - (permeat.p * permeat.Gas.yAr)));
f2wC2H4: (permeat.nvolflow/3600) * permeat.Gas.yC2H4 = PC2H4 * Am
    * (((feed.p * feed.Gas.yC2H4) - (permeat.p * permeat.Gas.yC2H4)
    ));
                                    .
                                    .
                                    .
f2wO2:  (permeat.nvolflow/3600) * permeat.Gas.yO2 = PO2 * Am * (((
    feed.p * feed.Gas.yO2) - (permeat.p * permeat.Gas.yO2)));
f2wSO2: (permeat.nvolflow/3600) * permeat.Gas.ySO2 = PSO2 * Am *
    (((feed.p * feed.Gas.ySO2) - (permeat.p * permeat.Gas.ySO2)));
```

Additionally to Code 4.170 all the 18 equations (Code 4.171) for each gaseous component are required to define the retentate mass flow. It is assumed that all the permeances are known.

Code 4.172: Equations to calculate the pressure values

```
fpress1:        feed.p = permeat.p + dp_permeat;
fpress2:        feed.p = retentate.p + dp_retentate;
```

Code 4.173: Equations to calculate the temperature values

```
ftemp1: feed.t = permeat.t + dt_permeat;
ftemp2: feed.t = retentate.t + dt_retentate;
```

Code 4.174: Equations to calculate the energy balance

```
fQ_loss: feed.massflow*(feed.lhv_total+feed.q_298_total) =
    retentate.massflow*(retentate.lhv_total+retentate.q_298_total)+
    permeat.massflow*(permeat.q_298_total+permeat.lhv_total)+Q_loss
    *3600;
fq_loss: Q_loss * 3600.0 = feed.massflow * feed.h * q_loss_rel /
    100.0;
fExergy: permeat.Exergy + retentate.Exergy + E_loss = feed.Exergy;
```

For the model validation an energy and mass balance have been made with experimental data from a membrane module, operated for the simple hydrogen project at the CHP plant Oberwart (Table 4.26/4.27).

Energy balance membrane		
Feed	8671.7	kW
Permeate	1872.4	kW
Retentate	6785.2	kW
Q_Loss	14.051	kW
Sum	0.0	kW

Table 4.26: Energy balance of membrane module

Mass balance membrane		
Feed	2164.9	kg/h
Permeate	323.46	kg/h
Retentate	1841.4	kg/h
Sum	0.0	kg/h

Table 4.27: Mass balance of membrane module

	CH_4	H_2	CO	CO_2	Sum
Measured feed	7.7	44.8	21.5	24.4	98.40
Simulation feed	8.23	45.67	20.27	21.81	95.98
Measured retentate	10.8	28.8	29.7	26.3	95.60
Simulation retentate	11.38	31.68	27.55	23.74	94.35
Measured permeate	0.4	82.4	2.3	19.8	104.90
Simulation permeate	0.371	80.5	2.12	17.02	100.01

Table 4.28: Comparison of measured and simulated Feed, Retentate and Permeate flows in vol.%

In Table 4.28 a comparison of simulated data with measured data from Oberwart is presented. The measured permances of CH_4, H_2, CO, CO_2 are taken from the dissertation of David Konlechner [31]. The content of the remaining other species is set to zero in the permeate stream. As a result, the retentate stream carries 100% of the remaining species and a share of the four mentioned main components.

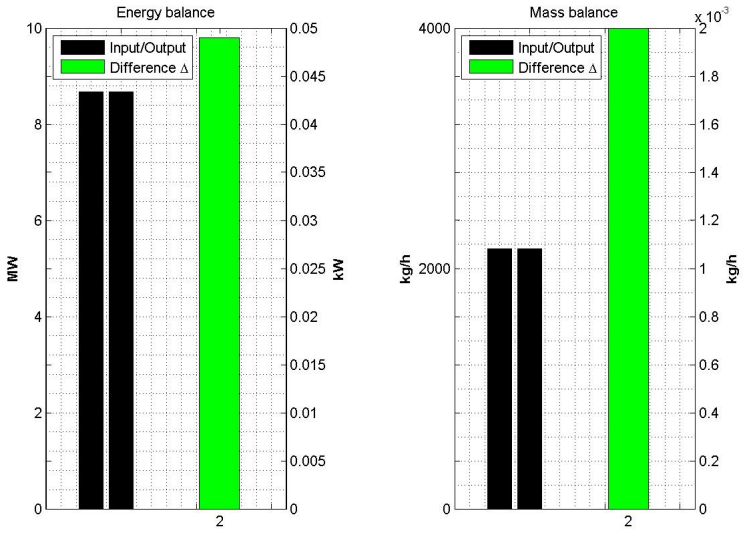


Figure 4.28: Energy and mass balance for the membrane

The values of Figure 4.28 are obtained by an simulation approach for the Oberwart power plant. Neither for the energy nor mass balance, there is a meaningful *Difference* Δ between input and output.

Evaporative cooler u_cooltwr_wet_

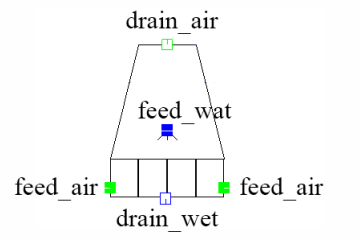


Figure 4.29: Evaporative cooler library icon

The evaporative cooler icon has two *feed_air* entrances. Air can either be fed from the left or from the right side, as it fits best for the simulation. The air is released via the outlet *drain_air*. The air outlet has to have a different gas global, as its moisture content will change in comparison to the inlet. The water to be cooled is entering through the entrance *feed_wat* and leaves at the connection *drain_wat*.

Variables	Description
mass_loss	Percentage lost by the evaporation of the cooling water [%]
dp_water	Pressure drop of the media to be cooled [bar]
Q_trans	Transfered heating power [kW]
T_sat_ambient	Saturation temperature of the ambient air [°C]
dt_ambient	Temperature difference of the exiting cooling water to the ambient temperature [°C]
E_loss	Exergy loss of the unit [kW]

Table 4.29: Evaporative cooler model item description

Code 4.175: Energy balances

```
f_ennergy:      feed_wat.massflow*feed_wat.h_total + feed_air.
    massflow * feed_air.h_total = drain_wat.massflow * drain_wat.
    h_total + drain_air.massflow * drain_air.h_total;
```

Code 4.175 shows the overall energy balance of the cooler.

Code 4.176: Heat transferred

```
f_ennergy:      feed_wat.massflow*feed_wat.h_total + feed_air.
    massflow * feed_air.h_total = drain_wat.massflow * drain_wat.
    h_total + drain_air.massflow * drain_air.h_total;
```

The transferred energy *Q_trans* to cool the liquid media is calculated by the equation of Code 4.176.

Code 4.177: Pressure loss air

```
f_p_ambeint:    feed_air.p = drain_air.p;
```

It is assumed that no pressure loss occurs for the aspirated ambient air. The assumption is implemented by Code 4.177.

Code 4.178: Mass balances

```
fmass_H2O:      drain_air.massflow*drain_air.Gas.wH2O + drain_wat.
    massflow = feed_air.massflow*feed_air.Gas.wH2O + feed_wat.
    massflow;
fmass_Ar:       drain_air.massflow*drain_air.Gas.wAr = feed_air.
    massflow*feed_air.Gas.wAr;
fmass_C2H4:     drain_air.massflow*drain_air.Gas.wC2H4 = feed_air.
    massflow*feed_air.Gas.wC2H4;
                                .
                                .
                                .
fmass_SO2:      drain_air.massflow*drain_air.Gas.wSO2 = feed_air.
    massflow*feed_air.Gas.wSO2;
```

A correct mass balance for the cooler is guaranteed due to the individual mass balance for each species (Code 4.178). Water is present in the air and of course the water stream. All the other species do not leave the gaseous stream.

Code 4.179: Water loss

```
f_mass:          drain_wat.massflow = feed_wat.massflow * (1.0 -
    mass_loss / 100.0);
```

Due to evaporation, there is a loss of water for the liquid stream. The loss is defined in percent in Code 4.179.

Code 4.180: Pressure drop water

```
f_pres:         feed_wat.p = drain_wat.p + dp_water;
```

Code 4.181: Pressure drop between air inlet and water outlet

```
f_drain_wat:    drain_wat.p = feed_air.p;
```

The pressure drop between water inlet and outlet is obtained by Code 4.180, while the pressure drop of air inlet and water outlet is set to zero by Code 4.181.

Code 4.182: Relative humidity

```
fphirel:        drain_air.Gas.yH2O*drain_air.p = feed_wat.wfp0t(
    drain_air.t);
```

The relative humidity is assumed to be 100% for the exiting gas stream (Code 4.182). The principle of Code 4.182 is the same as described for Code 4.86 and Equation 4.7.

Code 4.183: Ambient saturation temperature

```
fT_sat:         feed_air.p * feed_air.Gas.yH2O =   feed_wat.wfp0t(
    T_sat_ambient);
```

The ambient saturation temperature is calculated by Code 4.183. All other variables are known but *T_sat_ambient*.

Code 4.184: Temperature difference of exiting water to ambient air

```
fdt_ambienb:      dt_ambient = drain_wat.t - feed_air.t;
```

The difference of the water outlet and the ambient air inlet temperature is defined as *dt_ambient* and obtained by Code 4.184.

Code 4.185: Exergy loss

```
f_E_loss:         E_loss + drain_wat.Exergy + drain_air.Exergy =
    feed_air.Exergy + feed_wat.Exergy;
```

The exergy loss E_loss is calculated via Code 4.185. Figure 4.30 shows, as for any other

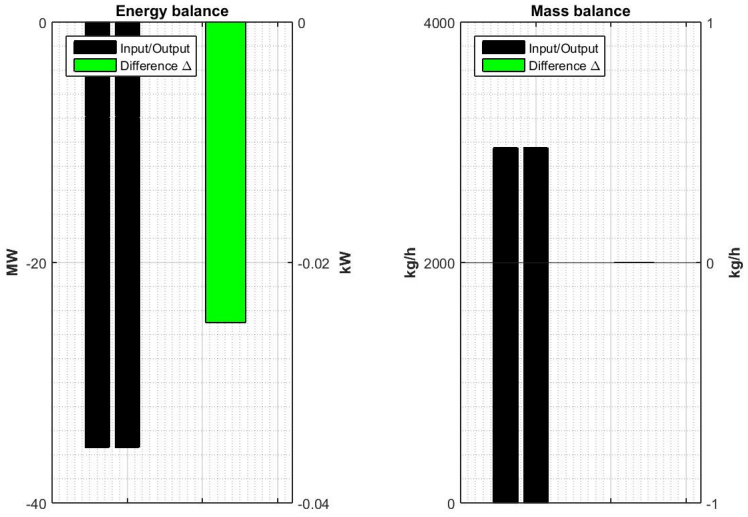


Figure 4.30: Energy and mass balance for the evaporation cooler

unit, the mass and energy balance which fulfilled besides an insignificant deviation due to an iteration errors.

Flue gas denitrification (DeNOx) scrubber u_denox_scr_

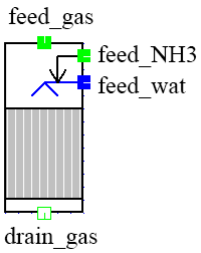


Figure 4.31: DeNOx scrubber library icon

The DeNOx scrubber is a model of a catalytic scrubber. All nitrogen compounds are transferred to N_2 as well as traces of NO and NH_3 in the clean gas. The gas stream enters at the *feed_gas* connection and leaves at the outlet called *drain_gas*. Due to the inlet *feed_NH3*, ammonia is fed into the scrubber. In practice the fed NH_3 is dissolved in water, therefore also water is fed into the scrubber by the connection *feed_wat*. By conversion rates the separation is defined.

Variables	Description
dp_gas	Pressure drop of gas [bar]
dp_wat	Pressure drop of the media to be cooled [bar]
massflow_liq	Mass flow of injected liquid (ammonia + water) [kg/h]
inject_rate	Amount of liquid injected per amount of gas [g/Nm^3]
NH3_conc	Ammonia concentration in the injected water (mass of NH_3 / total mass) [wt%]
O2_ref	Reference O_2 concentration in clean gas for NO_2 and NH_3 values [vol.%]
NO2_clean_gas	NO as NO_2 in dry fraction of clean gas corrected to 11% O_2 [g/Nm^3]
NH3_clean_gas	NH_3 in dry fraction of clean gas corrected to 11% O_2 [g/Nm^3]
NO_conv	Conversion of NO in the scrubber [%]
NH3_conv	Conversion of NH_3 in scrubber [%]
E_loss	Exergy loss [kW]

Table 4.30: DeNOx scrubber model item description

Code 4.186: Energy balances

```
fh_total:        drain_gas.massflow*drain_gas.h_total = feed_gas.
    massflow*feed_gas.h_total + feed_wat.massflow*feed_wat.h_total
    + feed_NH3.massflow*feed_NH3.h_total;
```

The energy balance of the scrubber is listed in Code 4.186.

Code 4.187: Pressure drop of gas stream

```
fdp_gas:         drain_gas.p + dp_gas = feed_gas.p;
```

Code 4.188: Pressure drop of injected liquid

```
fdp_wat:        drain_gas.p + dp_wat = feed_wat.p;
```

Code 4.189: Pressure gaseous ammonia

```
fdp_nh3:        feed_wat.p = feed_NH3.p;
```

The gaseous and liquid pressure drops are defined by Code 4.187 and 4.188. The inlet pressure of the water and ammonia stream are defined as equal by Code 4.189, so are their temperatures via Code 4.190.

Code 4.190: Pressure gaseous ammonia

```
fdp_nh3:        feed_wat.t = feed_NH3.t;
```

Code 4.191: Sum of injected scrubbing agent

```
finject_rate:   inject_rate*feed_gas.nvolflow = 1000.0*
    massflow_liq;
```

The sum of ammonia and water is presented by the variable *massflow_liq* and calculated after Code 4.191. Further, the *injection_rate* for the scrubbing agent is defined by Code 4.192.

Code 4.192: Injection rate

```
finject_rate:   inject_rate*feed_gas.nvolflow = 1000.0*
    massflow_liq;
```

Code 4.193: Relation between injected water and introduced gaseous ammonia

```
finject_rate:   inject_rate*feed_gas.nvolflow = 1000.0*
    massflow_liq;
```

Another characteristic value is the mass flow ratio between injected water and formally introduced gaseous ammonia *NH3_conc*, calculated through Code 4.193.

Code 4.194: Elementary mass balances

```
fAr:    drain_gas.massflow*drain_gas.Gas.wAr = feed_gas.massflow*
    feed_gas.Gas.wAr;

fC:     drain_gas.massflow*12.011*(2*drain_gas.Gas.wC2H4/28.0536 +
     2*drain_gas.Gas.wC2H6/30.0694 + 3*drain_gas.Gas.wC3H8/44.0962
    + drain_gas.Gas.wCH4/16.0428 + drain_gas.Gas.wCO/28.0104 +
    drain_gas.Gas.wCO2/44.0098 + drain_gas.Gas.wHCN/27.02568) =
    feed_gas.massflow*12.011*(2*feed_gas.Gas.wC2H4/28.0536 + 2*
    feed_gas.Gas.wC2H6/30.0694 + 3*feed_gas.Gas.wC3H8/44.0962 +
    feed_gas.Gas.wCH4/16.0428
+ feed_gas.Gas.wCO/28.0104 + feed_gas.Gas.wCO2/44.0098 + feed_gas.
    Gas.wHCN/27.02568);
                                        .
                                        .
```

```
fCl:    drain_gas.massflow*35.4527*drain_gas.Gas.wHCl/36.46064
        = feed_gas.massflow*35.4527*feed_gas.Gas.wHCl/36.46064;
```

Code 4.194 describes the mass balances for the elements Ar, C, H, O, N, S, Cl to generate the drain gas composition (Code 4.194).

Code 4.195: Inert hydrocarbons

```
fr_C2H4:        drain_gas.nvolflow*drain_gas.Gas.yC2H4 = feed_gas.
    nvolflow*feed_gas.Gas.yC2H4;
fr_C2H6:        drain_gas.nvolflow*drain_gas.Gas.yC2H6 = feed_gas.
    nvolflow*feed_gas.Gas.yC2H6;
fr_C3H8:        drain_gas.nvolflow*drain_gas.Gas.yC3H8 = feed_gas.
    nvolflow*feed_gas.Gas.yC3H8;
fr_CH4: drain_gas.nvolflow*drain_gas.Gas.yCH4 = feed_gas.nvolflow*
    feed_gas.Gas.yCH4;
```

It is assumed that no reaction of hydrocarbons occurs (Code 4.195).

Code 4.196: Further inert species

```
fr_CO:  drain_gas.nvolflow*drain_gas.Gas.yCO = feed_gas.nvolflow*
    feed_gas.Gas.yCO;
fr_H2:  drain_gas.nvolflow*drain_gas.Gas.yH2 = feed_gas.nvolflow*
    feed_gas.Gas.yH2;
fr_H2S: drain_gas.nvolflow*drain_gas.Gas.yH2S = feed_gas.nvolflow*
    feed_gas.Gas.yH2S;
fr_HCN: drain_gas.nvolflow*drain_gas.Gas.yHCN = feed_gas.nvolflow*
    feed_gas.Gas.yHCN;
fr_N2O: drain_gas.Gas.yN2O = 0.0;
fr_NO:  NO2_clean_gas*(1.0 - drain_gas.Gas.yH2O) = drain_gas.Gas.
    yNO*46.00554*44617.5*(O2_ref/100.0 - 0.21)/(drain_gas.Gas.yO2
    /(1.0 - drain_gas.Gas.yH2O) - 0.21);        # See Heft
    04/20.06.2003
fr_NH3: NH3_clean_gas*(1.0 - drain_gas.Gas.yH2O) = drain_gas.Gas.
    yNH3*17.03056*44617.5*(O2_ref/100.0 - 0.21)/(drain_gas.Gas.yO2
    /(1.0 - drain_gas.Gas.yH2O) - 0.21);        # See Heft
    04/20.06.2003
fr_NOx: feed_gas.nvolflow*(feed_gas.Gas.yNH3 - feed_gas.Gas.yNO) +
     feed_NH3.nvolflow*(feed_NH3.Gas.yNH3 - feed_NH3.Gas.yNO) =
    drain_gas.nvolflow*(drain_gas.Gas.yNH3 - drain_gas.Gas.yNO);
```

Also the species of Code 4.196 are assumed to not react. The drain gas content of CO_2, H_2O and HCl are calculated due to the elementary mass balance (Code 4.194).

Code 4.197: Nitrogen compound reduction to N_2

```
fr_N2O: drain_gas.Gas.yN2O = 0.0;
fr_NO:  NO2_clean_gas*(1.0 - drain_gas.Gas.yH2O) = drain_gas.Gas.
    yNO*46.00554*44617.5*(O2_ref/100.0 - 0.21)/(drain_gas.Gas.yO2
    /(1.0 - drain_gas.Gas.yH2O) - 0.21);        # See Heft
    04/20.06.2003
fr_NH3: NH3_clean_gas*(1.0 - drain_gas.Gas.yH2O) = drain_gas.Gas.
    yNH3*17.03056*44617.5*(O2_ref/100.0 - 0.21)/(drain_gas.Gas.yO2
```

```
    /(1.0 - drain_gas.Gas.yH2O) - 0.21);      # See Heft
    04/20.06.2003
fr_NOx: feed_gas.nvolflow*(feed_gas.Gas.yNH3 - feed_gas.Gas.yNO) +
     feed_NH3.nvolflow*(feed_NH3.Gas.yNH3 - feed_NH3.Gas.yNO) =
    drain_gas.nvolflow*(drain_gas.Gas.yNH3 - drain_gas.Gas.yNO);
```

Nitrogen components are reduced to N_2 as it is modeled by the equations of Code 4.197. For the simulation in PSE it has to be mentioned that the NO and NH_3 of the drain gas composition have to be set by the user.

Code 4.198: NOx conversion

```
fr_NOx: feed_gas.nvolflow*(feed_gas.Gas.yNH3 - feed_gas.Gas.yNO) +
     feed_NH3.nvolflow*(feed_NH3.Gas.yNH3 - feed_NH3.Gas.yNO) =
    drain_gas.nvolflow*(drain_gas.Gas.yNH3 - drain_gas.Gas.yNO);
```

Code 4.199: NH_3 conversion

```
fNH3_conv:      drain_gas.nvolflow*drain_gas.Gas.yNH3 = (1.0 -
    NH3_conv/100.0)*(feed_gas.nvolflow*feed_gas.Gas.yNH3 + feed_NH3
    .nvolflow*feed_NH3.Gas.yNH3);
```

The characteristic values for NOx and NH_3 conversion are generated by Code 4.198 and 4.199.

Code 4.200: Exergy loss

```
fE_loss:        drain_gas.Exergy + E_loss = feed_gas.Exergy +
    feed_NH3.Exergy + feed_wat.Exergy;
```

The loss of exergy is calculated by Code 4.200. Figure 4.32 shows the mass and energy balance of this unit.

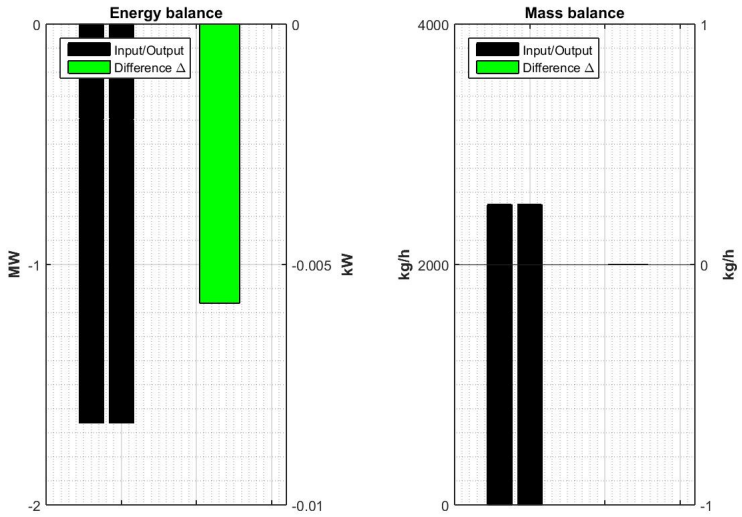


Figure 4.32: Energy and mass balance for the DeNOx scrubber

Drum for liquid/steam separation in a water stream u_drum_w_

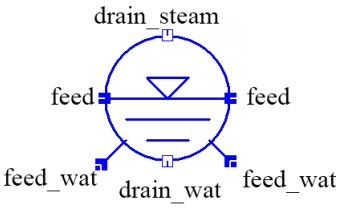


Figure 4.33: Liquid-steam separation drum library icon

This model splits unsaturated steam into saturated steam and liquid water. The water/steam mixture enters at the *feed* connection. Feed-water can be fed into the drum as well (*feed_wat*). For a steam power plant the feed-water is preheated in the economizer. Flue gas is streaming through the economizer -at a low temperature level- after passing the evaporator, superheater and reheater. Just to clarify the scenario. Additional steam is created by sending feed-water into the drum while the water level almost remains constant. But as already mentioned the separation drum can also be operated without preheated feed-water injection. Both, the *feed* and *feed_wat* can either be connected from the left or right side, as individually required. Liquid water is removed from the drum via the exit *drain_wat*. The steam stream leaves at the connection *drain_steam*.

Variables	Description
dp_steam	Pressure drop of gas [bar]
dp_wat	Pressure drop of the media to be cooled [bar]
dp_feed_wat	Pressure drop of the feed water [bar]
circ_ratio	Circulation ratio of the evaporating circuit: Ratio of the mass flow in the evaporating circuit referred to the produced steam [-]
E_loss	Exergy loss of the steam drum [kW]

Table 4.31: Separation drum model item description

Code 4.201: Energy balances

```
ifl ref(feed_wat) then
    fEnergy_feed_wat: drain_steam.massflow*drain_steam.h +
        drain_wat.massflow*drain_wat.h = feed.massflow*feed.h +
        feed_wat.massflow*feed_wat.h;
endifl
ifl !ref(feed_wat) then
        fEnergy: drain_steam.massflow*drain_steam.h +
            drain_wat.massflow*drain_wat.h = feed.massflow*feed.h;
endifl
```

Code 4.201, the energy balance for both cases, either supplying preheated feed-water to the drum or not.

Code 4.202: Mass balances

```
ifl ref(feed_wat) then
    fmassflow_wat:      drain_steam.massflow + drain_wat.massflow
       = feed.massflow + feed_wat.massflow;
endifl
ifl !ref(feed_wat) then
        fmassflow:      drain_steam.massflow + drain_wat.massflow
           = feed.massflow;
endifl
```

Also Code 4.202 shows both cases, with and without additional feed-water, this time for the mass balance.

Code 4.203: Pressure drop steam

```
fdp_steam:      drain_steam.p + dp_steam = feed.p;
```

Code 4.204: Pressure drop water

```
fdp_wat:        drain_wat.p + dp_wat = feed.p;
```

Code 4.205: Pressure drop feed-water

```
ifl ref(feed_wat) then
        fsp_feed_wat:  drain_wat.p + dp_feed_wat = feed_wat.p;
endifl
ifl !ref(feed_wat) then
        fsp_feed_wat_no:  dp_feed_wat = 0.0;
endifl
```

Code 4.203 and 4.204 calculate the pressure drop for the steam and water exit streams, related to the *feed* stream. Code 4.205 gains the pressure drop between feed-water input and the recirculating feed-water output (*drain_wat*). If the *feed_wat* inlet is not connected, the pressure drop is set to zero.

Code 4.206: Steam quantities

```
fx_steam:       drain_steam.x_steam = 1.0;
fx_wat: drain_wat.x_steam = 0.0;
```

Steam quantities for the drain water and gas stream are defined by Code 4.206. It is assumed that no water droplets are transported in the steam stream to the superheater. This is important as the post evaporation of water droplets in the superheater could lead to deposition of non evaporative components.

Code 4.207: Water/steam ratio

```
f_circ_ratio:   circ_ratio = drain_wat.massflow / drain_steam.
    massflow;
```

The variable *circ_ratio* describes the ratio between water and steam outlet mass flow. It is calculated according to Code 4.207. Also for the balance of exergy two separate cases are considered (Code 4.208).

Code 4.208: Exergy loss

```
if1 ref(feed_wat) then
        f1E_loss:       drain_wat.Exergy + drain_steam.Exergy +
            E_loss = feed.Exergy + feed_wat.Exergy;
endif1
if1 !ref(feed_wat) then
        f2E_loss:       drain_wat.Exergy + drain_steam.Exergy +
            E_loss = feed.Exergy;
endif1
```

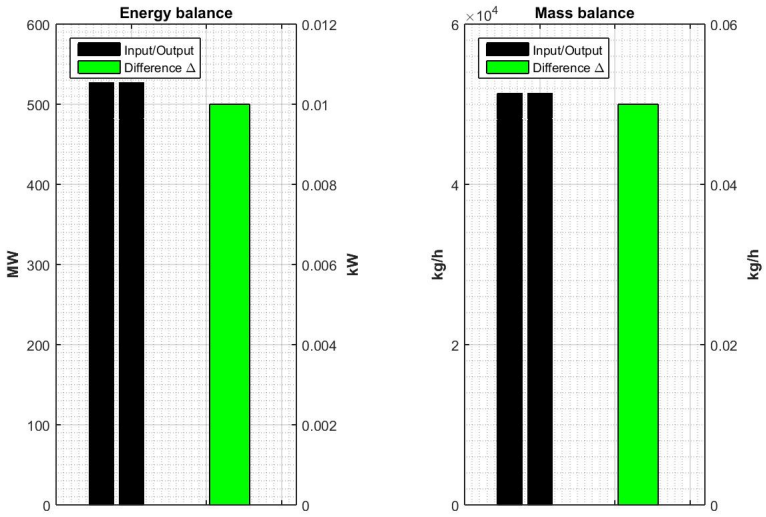


Figure 4.34: Energy and mass balance for the liquid-steam separation drum

Figure 4.34 presents the energy and mass balance of the drum unit.

Dryer u_dryer_ogw_

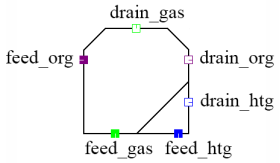


Figure 4.35: Dryer library icon

The model describes a dryer with direct or indirect heating. Biomass is fed through the inlet *feed_org*. It exits at *drain_org* without changing its composition except its water content. The heating medium for indirect drying is hot water, entering at *feed_htg* and leaving at *drain_htg*. For direct drying a gas stream enters at the inlet called *feed_gas* and leaves at the outlet *drain_gas*. Both options heating and drying are compulsory. If both are applied there is a combined heating. Additional theoretical and explanatory information can be found in [76][5.1.6].

Variables	Description
dp_steam	Pressure drop of the gas stream [bar]
dp_hw	Pressure drop of the attached indirect heating water [bar]
dt_org	This is the temperature difference between drain and feed organic stream [bar]
dt_gas_feed_org_dra	Temperature difference between gas feed and organic drain (relevant for counter-current dryers) [°C]
P_therm	Power used for vaporizing the water in the organic stream [kW]
Q_loss	Heat loss to the environment [kW]
q_loss_rel	This is the heat loss in percent according to the total drying power [%]
phi_rel_drain	Relative humidity in drain gas [%]
wfp0t	Returns vapour pressure of water at a certain temperature [bar]
E_loss	Exergy loss [kW]

Table 4.32: Dryer model item description

Code 4.209: Energy balances

```
ifl ref(feed_htg) && ref(feed_gas) then
fhtotal_c_h: drain_htg.massflow * drain_htg.h_total + drain_org.
    massflow * drain_org.h_total + drain_gas.massflow * drain_gas.
    h_total + Q_loss*3600 =
feed_htg.massflow * feed_htg.h_total + feed_org.massflow *
    feed_org.h_total + feed_gas.massflow * feed_gas.h_total;
endifl

ifl !ref(feed_htg) && ref(feed_gas) then
fhtotal_direct_h: drain_gas.massflow * drain_gas.h_total +
    drain_org.massflow * drain_org.h_total + Q_loss*3600 = feed_gas
```

```
    .massflow * feed_gas.h_total + feed_org.massflow * feed_org.
    h_total;
endifl

ifl ref(feed_htg) && !ref(feed_gas) then
fhtotal_ind_h:  drain_htg.massflow * drain_htg.h_total + drain_org
    .massflow * drain_org.h_total + drain_gas.massflow * drain_gas.
    h_total + Q_loss*3600 =
feed_htg.massflow * feed_htg.h_total + feed_org.massflow *
    feed_org.h_total;
endifl
```

In Code 4.209, at first the energy balance for combined heating is listed followed by direct heating and in the end the balance for indirect heating without a gaseous feed stream is listed.

Code 4.210: Mass balance of heating water

```
ifl ref(feed_htg) then
fmassheating:   feed_htg.massflow = drain_htg.massflow;
endifl
```

The entering and outgoing water mass is equal, see Code 4.210.

Code 4.211: Mass balance of gaseous components except steam

```
  ifl ref(feed_gas) then
fmassgcAr: drain_gas.massflow * drain_gas.Gas.wAr       = feed_gas
    .massflow * feed_gas.Gas.wAr;
fmassgcC2H4: drain_gas.massflow * drain_gas.Gas.wC2H4   = feed_gas
    .massflow * feed_gas.Gas.wC2H4;
                          .
                          .
fmassgcSO2: drain_gas.massflow * drain_gas.Gas.wSO2 = feed_gas.
    massflow * feed_gas.Gas.wSO2;
elsel
fmassgcAr_nofeed:       drain_gas.Gas.wAr = 0;
fmassgcC2H4_nofeed:     drain_gas.Gas.wC2H4 = 0;
                          .
                          .
fmassgcSO2_nofeed:      drain_gas.Gas.wSO2 = 0;
endifl
```

Code 4.211 ensures a correct species balance for direct heating. If there is no *feed_gas* connected to the model, the drain_gas only consists of water. This is defined in the source code after the *elsel* function. Water is not considered because even if no *feed_gas* is fed, water is removed from biomass via the *drain_gas* stream, due to other heating.

Code 4.212: Mass balance of water

```
ifl ref(feed_gas) then
fmassH2Oorggas1: drain_gas.massflow * drain_gas.Gas.wH2O +
    drain_org.massflow * drain_org.water_content = feed_gas.
    massflow * feed_gas.Gas.wH2O + feed_org.massflow * feed_org.
    water_content;
```

```
elsel
fmassH20orggas_nofeedgas: drain_gas.massflow * drain_gas.Gas.wH2O
    + drain_org.massflow * drain_org.water_content = feed_org.
    massflow * feed_org.water_content;
endifl
```

The water balance is considered separately in Code 4.212 due to its interaction between all the considered stream classes (organic, water, gas). The first balance is applied when gas is fed to the dryer, the second one if not.

Code 4.213: Mass balance of ash

```
ifl ref(feed_org.Ash) then
fmassAsh: drain_org.massflow*drain_org.ash_content = feed_org.
    massflow*feed_org.ash_content;
endifl
```

Code 4.214: Mass balance of organic components

```
fmass_org: (1.0 - drain_org.water_content - drain_org.ash_content
    ) * drain_org.massflow = (1.0 - feed_org.water_content -
    feed_org.ash_content) * feed_org.massflow;
```

The amount of ash is kept constant (Code 4.213). The mass balance for organic components is considered for water and ash free conditions (Code 4.214).

Code 4.215: Pressure drop gas stream

```
ifl ref(feed_gas) then
fdp_gas:drain_gas.p + dp_gas = feed_gas.p;
elsel
fdp_gas_nofeed: dp_gas = 0;
endifl
```

Code 4.216: Pressure drop heating water stream

```
ifl ref(feed_htg) then
        fdp_heating:    drain_htg.p + dp_hw = feed_htg.p;
elsel
        fdp_heating_no: dp_hw = 0;
endifl
```

For the gas (Code 4.215) and heating water (Code 4.216) streams, the pressure drops are calculated as presented as a first option. If one of the feed streams is not referenced, the pressure drop is set to zero.

Code 4.217: Temperature difference organic stream

```
fdt_org:        drain_org.t = feed_org.t + dt_org;
```

Code 4.218: Temperature difference gas feed and organic drain

```
fdt_gas_feed_org_drain: dt_gas_feed_org_drain = feed_gas.t-
    drain_org.t;
```

The temperature differences for the variables *dt_org* and *dt_gas_feed_org_drain* are results of Code 4.217 and 4.218.

Code 4.219: Pressure organic stream

```
fdp_org:        feed_org.p = drain_org.p;
```

Input and output pressure are equal (Code 4.219).

Code 4.220: Density of organic stream

```
forg_dens:      (1/feed_org.rho - feed_org.ash_content/feed_org.
    Ash.rho - feed_org.water_content/1000.0) / (1-feed_org.
    water_content-feed_org.ash_content) = (1/drain_org.rho -
    drain_org.ash_content/drain_org.Ash.rho - drain_org.
    water_content/1000.0) / (1-drain_org.water_content-drain_org.
    ash_content);
```

The density for the organic stream is calculated for water and ash free conditions (Code 4.220).

Code 4.221: Char balance gas stream

```
ifl ref(feed_gas.Char) then
fchar_cont:     drain_gas.nvolflow*drain_gas.char_content =
    feed_gas.nvolflow*feed_gas.char_content;
endifl
```

Code 4.222: Tar balance gas stream

```
ifl ref(feed_gas.Tar) then
ftar_cont: drain_gas.nvolflow*drain_gas.tar_content = feed_gas.
    nvolflow*feed_gas.tar_content;
endifl
```

Code 4.223: Dust balance gas stream

```
ifl ref(feed_gas.Dust) then
fdust_cont:     drain_gas.nvolflow*drain_gas.dust_content =
    feed_gas.nvolflow*feed_gas.dust_content;
endifl
```

Code 4.221, 4.222 and 4.223 show that no reactions for the mentioned components occur.

Code 4.224: Drying power

```
ifl !ref(feed_htg) then
        fP_therm_dh:    P_therm*3600.0 =        feed_gas.massflow*
            feed_gas.h + feed_gas.nvolflow*(feed_gas.dust_content
            /1000)*feed_gas.h_dust -
drain_gas.massflow*drain_gas.h - drain_gas.nvolflow*(drain_gas.
    dust_content/1000)*drain_gas.h_dust;
endifl

ifl ref(feed_htg) && ref(feed_gas) then
```

```
fP_therm_ch:    P_therm*3600.0  =       feed_gas.massflow*feed_gas
    .h + feed_gas.nvolflow*(feed_gas.dust_content/1000)*feed_gas.
    h_dust + feed_htg.massflow*feed_htg.h_total - drain_gas.
    massflow*drain_gas.h - drain_gas.nvolflow*(drain_gas.
    dust_content/1000)*drain_gas.h_dust - drain_htg.massflow*
    drain_htg.h_total;
endifl

ifl ref(feed_htg) && !ref(feed_gas) then
fP_therm_idh:   P_therm*3600.0 =        feed_htg.massflow*feed_htg
    .h_total - drain_htg.massflow*drain_htg.h_total;
endifl
```

The drying power is calculated for three different scenarios (Code 4.224). The first one represents direct heating, the second one combined heating and the last one indirect heating.

Code 4.225: Heat loss

```
        fQ_loss:        Q_loss = P_therm*q_loss_rel/100.0;
```

The heat loss *Q_loss* is determined in relation to the relative heat loss *q_loss_rel* (Code 4.225).

Code 4.226: Relative humidity of drain gas stream

```
fphi_rel_drain: (phi_rel_drain/100.0)*wfp0t(drain_gas.t) =
    drain_gas.Gas.yH2O*drain_gas.p;
```

The idea of a relative humidity of 100% is described for Code 4.86. The same approach is also applied for this code.

Code 4.227: Exergy loss

```
ifl ref(feed_htg) && ref(drain_htg) then
        f1E_loss:       drain_gas.Exergy + drain_org.Exergy +
            drain_htg.Exergy + E_loss = feed_htg.Exergy + feed_gas.
            Exergy + feed_org.Exergy;
elsel
        f2E_loss:       drain_gas.Exergy + drain_org.Exergy +
            E_loss = feed_gas.Exergy + feed_org.Exergy;
endifl
```

For the exergy loss it is distinguished between the case of combined and direct heating (Code 4.227).

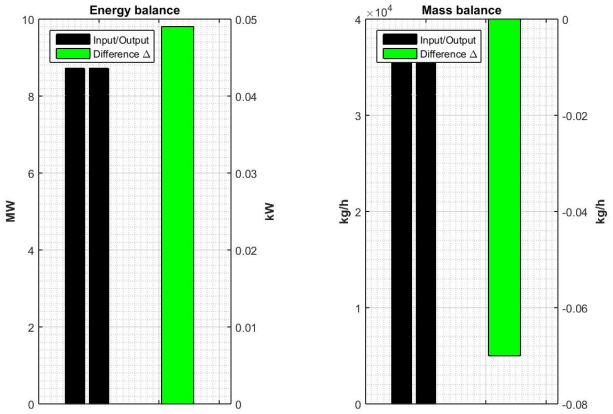


Figure 4.36: Energy and mass balance for the dryer

Figure 4.36 illustrates the mass and energy balance. The large difference bar is related to a different scale for the secondary ordinate.

Electric heater u_elht_g_

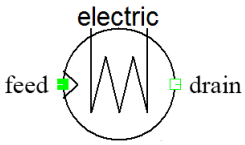


Figure 4.37: Electric heater library icon

The electric heater model simulates heating by electric power (Fig. 4.37). It is available for gaseous and water streams. All variables and connections are named equal for both stream classes. The inlet connection is called *feed* and the outlet *drain*. The code below is for the gaseous heater model.

Variables	Description
dp	Pressure drop of the gas stream [bar]
P_el	Electric power for heating the stream [kW]
q_loss_rel	Heat loss relative to sum of sensible heat in feed and electric power [%]
Q_loss	Heat loss of the electric heater [kW]
E_loss	Exergy loss [kW]

Table 4.33: Electric heater model item description

Code 4.228: Mass balances

```
fmassflow:      drain.massflow = feed.massflow;
ifl (ref(drain.Dust) && ref(drain.Dust)) then fmass_dust:
    drain.dust_content = feed.dust_content; endifl
ifl ref(drain.Char) && ref(drain.Char) then fmass_char: drain.
    char_content = feed.char_content; endifl
ifl ref(drain.Tar) && ref(drain.Tar) then fmass_tar:    drain.
    tar_content = feed.tar_content;   endifl
```

This code section is about the unit's mass balance (Code 4.228). The first line is for the overall mass balance. The following lines are representing individual cases if dust, char and/or tar are present in the gas stream, or not.

Code 4.229: Pressure drop

```
fdp:    feed.p - dp = drain.p;
```

The pressure drop of the gas stream is defined after Code 4.229.

Code 4.230: Energy balance

```
fh:     feed.massflow*feed.h_total/3600.0 + P_el =
drain.massflow*drain.h_total/3600.0 + Q_loss;
```

Code 4.230 shows the overall energy balance.

Code 4.231: Heat loss

```
fQ_loss: Q_loss = 0.01*q_loss_rel*(feed.massflow*feed.h/3600.0 +
    P_el);
```

Code 4.232: Exergy loss

```
f_E_loss:       E_loss = P_el + feed.Exergy - drain.Exergy;
```

Code 4.231 and 4.232 are for the calculation of the heat and exergy losses. Figure 4.38 shows the energy and mass balance.

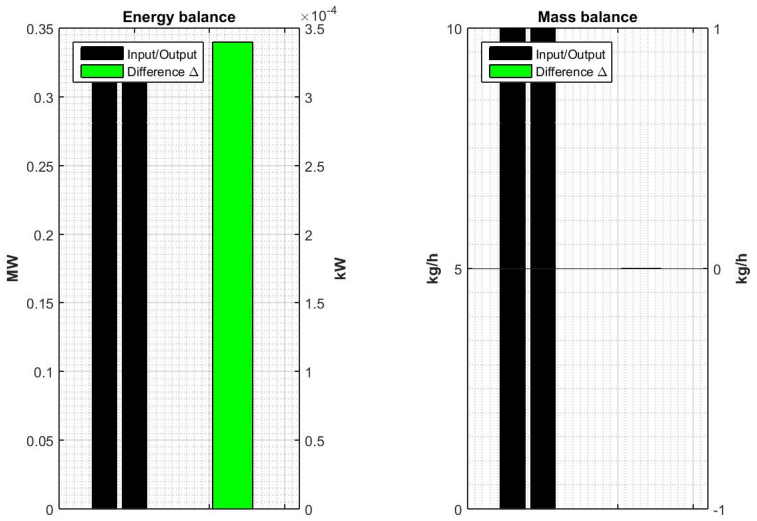


Figure 4.38: Energy and mass balance for the electric heater

Fuel cell u_fuelcell_g_

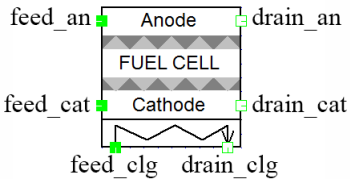


Figure 4.39: Fuel cell library icon

For the fuel cell simulation different models are available IRSOFC, IRSOFC_pol, MCFC_fuel_cell, PAFC_fuel_cell, SOFC. Three out of five are based on the solid oxide fuel cell (SOFC) technology. Their source codes differs just marginal from each other. Also the source codes of the molten carbonate fuel cell (MCFC) and the phosphoric acid fuel cell (PAFC) are following the same structure as the SOFC model. Therefore, the SOFC model is explained representative for all of them. The anode is fed with pure hydrogen by the connection *feed_an*. Due to oxidisation of hydrogen by O^{2-} from the cathode, water leaves the fuel cell at the *drain_an* connection. At the cathode oxygen enters at *feed_cat*, is reduced by electrons from the anode and forms O^{2-}. Not converted oxygen exits through the connection *drain_cat*. A gaseous cooling agent can be fed at feed_clg and leaves via drain_clg, but external cooling is not compulsory. Additional theoretical information on the fuel cell is given by T. Pröll [76][5.5].

Variables	Description
dp_an	Pressure drop of the anode flow over the fuel cell [bar]
dp_cat	Pressure drop of the cathode flow over the fuel cell [bar]
dp_clg	Pressure drop of optionally existing cooling gas, if no external cooling is connected, the pressure drop is set zero by the model [bar]
dt_an_cat	Temperature spread of exiting gas streams [°C]
eta_td	Thermodynamic efficiency of the fuel cell [%]
eta_u	Voltage efficiency of the fuel cell [%]
eta_i	Current efficiency of the fuel cell (X_fuel_total) [%]
eta_el	Electric efficiency of the fuel cell [%]
T_cell	Medium temperature of electrodes and electrolyte [°C]
E0	Theoretical open circuit voltage [V]
U_loss	Voltage loss owing to polarization effects (activation, concentration, ohmic, internal currents, fuel crossover) [V]
U	Actually reached electric potential of the operated fuel cell [V]
i	Specific current produced by the fuel cell [mA/cm^2]
i0	Reference current density for polarization loss calculation [mA/cm^2]
A	Active section area of the electrolyte [m^2]
P_el	Electric power of the fuel cell [kW]
q_loss_rel	Heat loss related to chemical fuel power P_chem [%]
X_fuel_total	Total fuel converted in terms of H_2 equivalents [%]
X_H2	Part of hydrogen converted on anode side [%]
X_CO	Part of carbon monoxide converted on anode side [%]
X_CH4	Part of methane converted on anode side [%]
X_C2H4	Part of C_2H_4 converted on anode side [%]
X_C2H6	Part of C_2H_6 converted on anode side [%]
X_C3H8	Part of C_3H_8 converted on anode side [%]
P_chem	Chemical fuel power of both anode and cathode feed, lhv-based [kW]
E_loss	Exergy loss of the fuel cell unit [kW]
fcf_E0	Function returns the open circuit voltage for the reaction H_2 + 0.5 O_2 → H_2O dependent on medium fuel cell temperature and partial pressures at gas entry and exit
Kp_CO_H2O_CO2_H2	Equilibrium constant of the water-gas-shift-reaction (CO + H_2O → CO_2 + H_2)

Table 4.34: Fuel cell model item description

Code 4.233: Energy balances

```
ifl ref(feed_clg) && ref(drain_clg) then
flh_total: drain_an.massflow*drain_an.h_total + drain_cat.massflow
    *drain_cat.h_total + drain_clg.massflow*drain_clg.h_total +
    3600.0*(P_el + Q_loss) = feed_an.massflow*feed_an.h_total +
    feed_cat.massflow*feed_cat.h_total
```

```
+ feed_clg.massflow*feed_clg.h_total;
elsel
f2h_total: drain_an.massflow*drain_an.h_total + drain_cat.massflow
    *drain_cat.h_total + 3600.0*(P_el + Q_loss) = feed_an.massflow*
    feed_an.h_total + feed_cat.massflow*feed_cat.h_total;
endifl
```

The first option of the if-condition describes the energy balance when external cooling is applied, the second one if not (Code 4.233).

Code 4.234: Chemical fuel power of both electrodes

```
fP_chem: P_chem*3.6 = feed_an.nvolflow*feed_an.Gas.LHV + feed_cat.
    nvolflow*feed_cat.Gas.LHV;
```

The chemical fuel power of both, anode and cathode feed stream, is calculated by Code 4.234.

Code 4.235: Thermodynamic efficiency of the fuel cell

```
feta_td: eta_td*0.01 = E0/1.2532;
```

Code 4.236: Voltage efficiency of the fuel cell

```
feta_u: eta_u*0.01 = U/E0;
```

Code 4.237: Current or conversion efficiency of the fuel cell

```
feta_i: eta_i = X_fuel_total;
```

Code 4.238: Electric efficiency of the fuel cell

```
feta_el: P_el = (eta_el/100.0)*P_chem;
```

Various efficiency coefficients, listed and described in Table 4.34, are obtained by Code 4.235 to 4.238.

Code 4.239: Cell temperature

```
fT_cell: T_cell = drain_cat.t;
```

It is assumed that the cell temperature, within the fuel cell stack, equals the cathode outlet temperature (Code 4.239).

Code 4.240: Theoretical open circuit voltage E0

```
fE0: E0 = fcf_E0 (T_cell, feed_an.p*feed_an.Gas.yH2, drain_an.p*
    drain_an.Gas.yH2, feed_cat.p*feed_cat.Gas.yO2, drain_cat.p*
    drain_cat.Gas.yO2, feed_an.p*feed_an.Gas.yH2O, drain_an.p*
    drain_an.Gas.yH2O);
```

Code 4.241: Polarisation losses

```
fU_loss:          U_loss = i/i0;
```

Code 4.242: Effective voltage

```
fU:        U + U_loss = E0;
```

Code 4.240 to 4.242 are calculation of characteristic values for fuel cell operation.

Code 4.243: Electric power of the fuel cell

```
fP_el:   P_el = U*(i*0.1)*A;
```

The electric power of the fuel cell is calculated through the effective voltage U, the total current i produced by the fuel cell and the active section area of the electrolyte A (Code 4.243).

Code 4.244: Heat loss related to P_chem

```
fq_loss_rel:      Q_loss = (q_loss_rel/100.0)*P_chem;
```

The heat loss is calculated according to Code 4.244.

Code 4.245: Anode side elementary mass balances

```
fAr_an: drain_an.massflow*drain_an.Gas.wAr = feed_an.massflow*
    feed_an.Gas.wAr;
fC_an:   drain_an.massflow*12.011*(2*drain_an.Gas.wC2H4/28.0536 +
    2*drain_an.Gas.wC2H6/30.0694 + 3*drain_an.Gas.wC3H8/44.0962 +
    drain_an.Gas.wCH4/16.0428 + drain_an.Gas.wCO/28.0104 + drain_an
    .Gas.wCO2/44.0098 + drain_an.Gas.wHCN/27.02568) = feed_an.
    massflow*12.011*(2*feed_an.Gas.wC2H4/28.0536 + 2*feed_an.Gas.
    wC2H6/30.0694 + 3*feed_an.Gas.wC3H8/44.0962 + feed_an.Gas.wCH4
    /16.0428 + feed_an.Gas.wCO/28.0104 + feed_an.Gas.wCO2/44.0098 +
     feed_an.Gas.wHCN/27.02568);
                                    .
                                    .
fCl_an: drain_an.massflow*35.4527*drain_an.Gas.wHCl/36.46064 =
    feed_an.massflow*35.4527*feed_an.Gas.wHCl/36.46064;
```

Code 4.246: Cathode side elementary mass balances

```
fAr_cat: drain_cat.massflow*drain_cat.Gas.wAr = feed_cat.massflow*
    feed_cat.Gas.wAr;
fC_cat:  drain_cat.massflow*12.011*(2*drain_cat.Gas.wC2H4/28.0536 +
     2*drain_cat.Gas.wC2H6/30.0694 + 3*drain_cat.Gas.wC3H8/44.0962
    + drain_cat.Gas.wCH4/16.0428 + drain_cat.Gas.wCO/28.0104 +
    drain_cat.Gas.wCO2/44.0098 + drain_cat.Gas.wHCN/27.02568) =
    feed_cat.massflow*12.011*(2*feed_cat.Gas.wC2H4/28.0536 + 2*
    feed_cat.Gas.wC2H6/30.0694 + 3*feed_cat.Gas.wC3H8/44.0962 +
    feed_cat.Gas.wCH4/16.0428 + feed_cat.Gas.wCO/28.0104 + feed_cat
    .Gas.wCO2/44.0098 + feed_cat.Gas.wHCN/27.02568);
                                    .
                                    .
fCl_cat: drain_cat.massflow*35.4527*drain_cat.Gas.wHCl/36.46064 =
    feed_cat.massflow*35.4527*feed_cat.Gas.wHCl/36.46064;
```

In combination with the following species balances Code 4.245 and 4.246 are responsible for the drain gas composition of both electrodes.

Code 4.247: Partial steam reforming of hydrocarbons

```
fr_C2H4_an: drain_an.nvolflow*drain_an.Gas.yC2H4 = (1.0 - X_C2H4
    /100.0)*feed_an.nvolflow*feed_an.Gas.yC2H4;
fr_C2H6_an: drain_an.nvolflow*drain_an.Gas.yC2H6 = (1.0 - X_C2H6
    /100.0)*feed_an.nvolflow*feed_an.Gas.yC2H6;
fr_C3H8_an: drain_an.nvolflow*drain_an.Gas.yC3H8 = (1.0 - X_C3H8
    /100.0)*feed_an.nvolflow*feed_an.Gas.yC3H8;
fr_CH4_an:  drain_an.nvolflow*drain_an.Gas.yCH4  = (1.0 - X_CH4
    /100.0)*feed_an.nvolflow*feed_an.Gas.yCH4;
```

Code 4.248: Hydrogen conversion

```
frH2_an: drain_an.nvolflow*drain_an.Gas.yH2 = (1.0 - X_H2/100.0)*
    feed_an.nvolflow*feed_an.Gas.yH2;
```

Code 4.249: Carbon monoxide conversion

```
frCO_an: drain_an.nvolflow*drain_an.Gas.yCO = (1.0 - X_CO/100.0)*
    feed_an.nvolflow*feed_an.Gas.yCO;
```

Code 4.250: Total fuel conversion in terms of H_2 equivalents

```
fX_fuel_total:  X_fuel_total/100.0      = 1.0 - (drain_an.nvolflow
    *(drain_an.Gas.yH2 + drain_an.Gas.yCO + 4.0*drain_an.Gas.yCH4 +
     6.0*drain_an.Gas.yC2H4 + 7.0*drain_an.Gas.yC2H6 + 10.0*
    drain_an.Gas.yC3H8))/(feed_an.nvolflow*(feed_an.Gas.yH2 +
    feed_an.Gas.yCO + 4.0*feed_an.Gas.yCH4 + 6.0*feed_an.Gas.yC2H4
    + 7.0*feed_an.Gas.yC2H6 + 10.0*feed_an.Gas.yC3H8));
```

Rates for hydro carbon, hydrogen and carbon monoxide conversion as well as the total fuel conversion -in terms of H_2 equivalents- are defined via Code 4.247 to 4.250.

Code 4.251: Inert components

```
fr_H2S_an: drain_an.nvolflow*drain_an.Gas.yH2S = feed_an.nvolflow*
    feed_an.Gas.yH2S;
fr_HCN_an: drain_an.nvolflow*drain_an.Gas.yHCN = feed_an.nvolflow*
    feed_an.Gas.yHCN;
fr_N2_an: drain_an.nvolflow*drain_an.Gas.yN2 = feed_an.nvolflow*
    feed_an.Gas.yN2;
fr_N2O_an: drain_an.nvolflow*drain_an.Gas.yN2O = feed_an.nvolflow*
    feed_an.Gas.yN2O;
fr_NH3_an: drain_an.nvolflow*drain_an.Gas.yNH3 = feed_an.nvolflow*
    feed_an.Gas.yNH3;
```

Via Code 4.251 it is ensured that H_2S, HCN, N_2, N_2O, NH_3 are considered as inert components.

Code 4.252: Oxygen anode side

```
fr_O2_an: drain_an.Gas.yO2 = 0.0;
```

It is assumed that all of eventually existing free oxygen on the anode side is reacting, so that none of it is present in the drain stream (Code 4.252).

Code 4.253: Cathode side reactions

```
fr_C2H4_cat: drain_cat.nvolflow*drain_cat.Gas.yC2H4 = feed_cat.
    nvolflow*feed_cat.Gas.yC2H4;
fr_C2H6_cat: drain_cat.nvolflow*drain_cat.Gas.yC2H6 = feed_cat.
    nvolflow*feed_cat.Gas.yC2H6;
fr_C3H8_cat: drain_cat.nvolflow*drain_cat.Gas.yC3H8 = feed_cat.
    nvolflow*feed_cat.Gas.yC3H8;
fr_CH4_cat: drain_cat.nvolflow*drain_cat.Gas.yCH4 = feed_cat.
    nvolflow*feed_cat.Gas.yCH4;
fr_CO_cat: drain_cat.nvolflow*drain_cat.Gas.yCO = feed_cat.
    nvolflow*feed_cat.Gas.yCO;
fr_H2_cat: drain_cat.nvolflow*drain_cat.Gas.yH2 = feed_cat.
    nvolflow*feed_cat.Gas.yH2;
fr_H2S_cat: drain_cat.nvolflow*drain_cat.Gas.yH2S = feed_cat.
    nvolflow*feed_cat.Gas.yH2S;
fr_HCN_cat: drain_cat.nvolflow*drain_cat.Gas.yHCN = feed_cat.
    nvolflow*feed_cat.Gas.yHCN;
fr_N2O_cat: drain_cat.nvolflow*drain_cat.Gas.yN2O = feed_cat.
    nvolflow*feed_cat.Gas.yN2O;
fr_NH3_cat: drain_cat.nvolflow*drain_cat.Gas.yNH3 = feed_cat.
    nvolflow*feed_cat.Gas.yNH3;
fr_NO_cat: drain_cat.nvolflow*drain_cat.Gas.yNO = feed_cat.
    nvolflow*feed_cat.Gas.yNO;
```

Code 4.253 indicates that all the listed gases pass the fuel cell without reacting. The species Ar, CO_2, H_2O, HCl, N_2 and SO_2 are not listed above as their content in the drain stream is already defined by the elementary balance of the cathode 4.246.

Code 4.254: Anode side pressure drop

```
fdp_an:          drain_an.p + dp_an = feed_an.p;
```

Code 4.255: Cathode side pressure drop

```
fdp_cat:         drain_cat.p + dp_cat = feed_cat.p;
```

Code 4.256: Temperature spread of exiting gas streams anode-cathode

```
fdt_an_cat:      drain_cat.t + dt_an_cat = drain_an.t;
```

Pressure drops and the temperature difference between anode and cathode drain stream are shown by Code 4.254 to 4.256.

Code 4.257: External cooling gas mass balance and pressure drop

```
ifl ref(feed_clg) && ref(drain_clg) then
        fmass_clg: drain_clg.massflow = feed_clg.massflow;
        f1dp_clg:        drain_clg.p + dp_clg = feed_clg.p;
elsel
        f2dp_clg: dp_clg = 0.0;
endifl
```

Code 4.258: External cooling gas mass balance and pressure drop

```
if1 ref(feed_clg) && ref(drain_clg) then
        f1E_loss: drain_an.Exergy + drain_cat.Exergy + drain_clg.
            Exergy + P_el + E_loss
                = feed_an.Exergy + feed_cat.Exergy + feed_clg.
                    Exergy;
else1
        f2E_loss: drain_an.Exergy + drain_cat.Exergy + P_el +
            E_loss
                = feed_an.Exergy + feed_cat.Exergy;
endif1
```

If external cooling is applied the first option of Code 4.257 and 4.258 is used otherwise option two. Figure 4.40 presents the energy and mass balance.

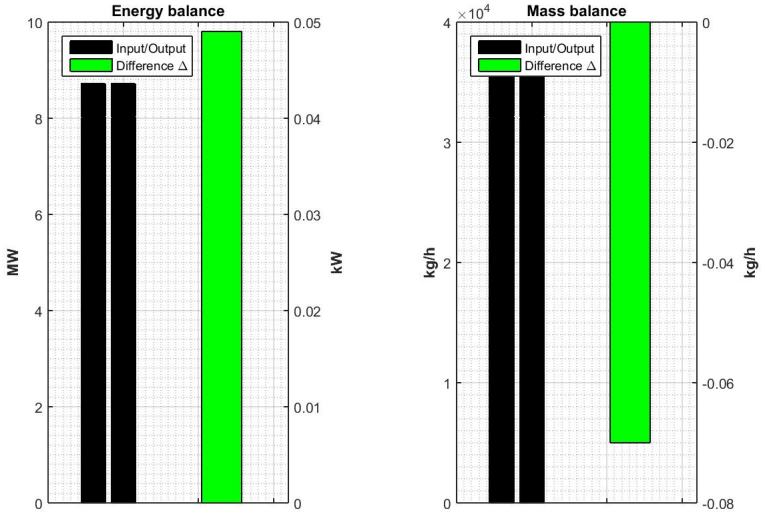


Figure 4.40: Energy and mass balance for the fuel cell

Gas engine u_gaseng2_

This model simulates a gas engine for electric power generation. Product gas or natural gas is fed via the *gas_feed* connector. The required combustion air is provided by the entrance *air_feed*. Exhaust gas leaves at the outlet connector *exhaust*. The shaft connector *shaft_in/shaft_out* provides the obtained mechanical energy to a generator unit. As always in case of shaft connections, the connector has to be defined as inlet and outlet connector, as described in Subsection 4.2.1. Theoretical information on the gas engine is provided by T. Pröll [76][5.4].

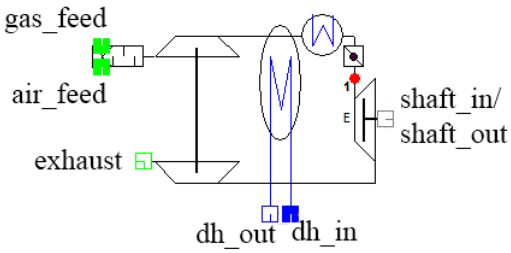


Figure 4.41: Gas engine library icon

Switch	Description
NOx_formation_model	Switches the formation of thermal NOx in the gas engine on, off or to a user defined mode

Variables	Description
dp_dh	Pressure drop district [bar]
P_therm	Calculates the energy content of the gaseouse fuel feed to the gas engine, LHV based [kW]
P_therm_nom	Nominal power of the gas engine, provided by the manufacture [kW]
lambda	Air excess ratio of gas engine: Standard value for wood gas combustion is 1.8 - 1.9 [kg/kg]
CO_slip	CO-slip of the engine. Part of CO of the gas mixture that slips uncombusted through the engine [%]
HC_slip	Hydrocarbon slip of the engine. Part of gaseous hydrocarbons of the gas mixture that slip uncombusted through the engine [%]
Tmix	Temperature in the gas mixer [°C]
Tmix_sat	Saturation temperature in the gas mixer [°C]
Tmix_diff_sat	Temperature difference to the saturation temperature [°C]
massflow_mix	Mass flow of the gas mixture [kg/h]
nvolflow_mix	Normal volume flow of gas mixture [Nm^3/h]
eta_mech	Mechanical efficicy of the gas engine [%]
P_shaft	Power transported through shaft [kW]
eta_therm	Efficiency for heat to district heating and ambient cooler (heat loss not included) - value provided by the engine type model [%]
Q_dh	Heat available for district heating [kW]

E_loss	Specifies the ambient conditions necessary for the gas engine (referencing to ambient global)
Globals	**Description**
q_loss_rel	Heat loss of the gas engine dependent on thermal power of the fuel (part of fuel power) [%]
Q_loss	Heat loss of the electric heater [kW]
E_loss	Exergy loss [kW]
Globals	**Description**
Gas_mix	Composition of the fuel gas and air mixture (referencing to gas global)

Table 4.35: Gas engine model item description

Code 4.259: Energy balance

```
fEnergy: exhaust.massflow*exhaust.h_total + (shaft_out.power +
    Q_dh + Q_loss)*3600.0 = gas_feed.massflow*gas_feed.h_total +
    air_feed.massflow*air_feed.h_total;
```

Code 4.260: Exergy loss

```
fExergy: exhaust.Exergy + dh_out.Exergy + shaft_out.power + E_loss
    = gas_feed.Exergy + air_feed.Exergy + dh_in.Exergy;
```

Code 4.259 and 4.260 show the overall energy and exergy balance for the gas engine model. Even though the shaft connector is also defined as an inlet, in these and the following code sections it is considered as an outlet. As the aim of a gas engine is to generate mechanical energy and not to consume it.

Code 4.261: Species balances

```
flwAr: massflow_mix * Gas_mix.wAr = gas_feed.massflow * gas_feed.
    Gas.wAr + air_feed.massflow * air_feed.Gas.wAr;
flwC2H4: massflow_mix * Gas_mix.wC2H4 = gas_feed.massflow *
    gas_feed.Gas.wC2H4  + air_feed.massflow * air_feed.Gas.wC2H4;
                        .
                        .
flwSO2: massflow_mix * Gas_mix.wSO2 = gas_feed.massflow * gas_feed
    .Gas.wSO2 + air_feed.massflow * air_feed.Gas.wSO2;
```

Code 4.262: Temperature of gas mixture

```
fmix:   gas_feed.h * gas_feed.massflow + air_feed.h * air_feed.
    massflow = gas_feed.Gas.gfht(Tmix) * gas_feed.massflow +
    air_feed.Gas.gfht(Tmix) * air_feed.massflow;
```

Code 4.263: Saturation temperature of gas mixture

```
tsat:   gas_feed.p * Gas_mix.yH2O = exp(62.1361 - 7258.2/(Tmix_sat
    +273.15) - 7.3037*ln(Tmix_sat+273.15) + 4.1653e-6*(Tmix_sat
    +273.15)^2);
```

Code 4.264: Difference between Tmix and Tmix_sat

```
f_t_diff_sat:    Tmix_diff_sat = Tmix - Tmix_sat;
```

The species balances of Code 4.261 creates the composition of the gas mixture of fuel gas plus combustion air. Its temperature *Tmix* is obtained by Code 4.262, the saturation temperature by Code 4.263 and the difference of both temperatures is calculated by Code 4.264.

Code 4.265: District heating energy balance

```
f_energy_dh:     Q_dh*3600.0 = dh_out.massflow*dh_out.h - dh_in.
    massflow*dh_in.h;
```

Code 4.266: District heating mass balance

```
f_mass_dh:       dh_out.massflow = dh_in.massflow;
```

Code 4.267: District heating pressure drop

```
f_dp_dh:         dh_in.p = dh_out.p + dp_dh;
```

Code 4.268: District heating output

```
f_q_therm:       Q_dh  * 100 = P_therm * eta_therm;
```

The energy balance for the waste heat usage is shown by Code 4.265, the associated mass balance by Code 4.266. It is assumed that the waste heat is applied for district heating. The pressure drop between in- and outlet stream is defined by Code 4.267. The relation between *Q_dh*, *P_therm* and *eta_therm* is set by Code 4.268.

Code 4.269: Heat loss

```
f_q_loss:        Q_loss * 100 = P_therm * q_loss_rel;
```

Code 4.269 generates the overall heat loss of the gas engine unit.

Code 4.270: Thermal power of the incoming fuel gas

```
ifl ref(gas_feed.Tar) then
fP_therm_tar: P_therm*3600.0 = gas_feed.nvolflow*(gas_feed.Gas.LHV
    *1000.0 + 0.001*gas_feed.tar_content*gas_feed.Tar.lhv) +
    air_feed.nvolflow*air_feed.Gas.LHV*1000.0;
elsel
fP_therm: P_therm*3600.0 = gas_feed.nvolflow *gas_feed.Gas.LHV *
    1000 + air_feed.nvolflow * air_feed.Gas.LHV *1000;
endifl
```

The thermal power of the input fuel is based on the lower heating value (LHV), see Code 4.270.

Code 4.271: Load factor

```
fload_factor:    P_therm * 100 = P_therm_nom * load_factor;
```

Code 4.272: Mechanical efficiency of the gas engine

```
f_eta_mech:      P_therm*eta_mech  = shaft_out.power*100.0;
```

Code 4.273: Mechanical power output of the gas engine

```
fP_shaft:        P_shaft =  shaft_out.power;
```

The characteristic values *load_factor*, *eta_mech* and *P_shaft* are essential to compare the simulated engines to other engines (Code 4.271 to 4.273).

Code 4.274: Air ratio lambda

```
flambda:         lambda*exhaust.massflow*(exhaust.Gas.wO2 - exhaust
    .Gas.wCO*31.9988/56.0208) = (lambda - 1.0)*(air_feed.massflow*
    air_feed.Gas.wO2 + gas_feed.massflow*gas_feed.Gas.wO2);
```

The air ratio lambda is defined as the actually available amount of oxygen divided by the stoichiometric required amount, see Code 4.274.

Code 4.275: Standard volume flow of gas mixture

```
fnvolflow_mix:   nvolflow_mix = gas_feed.nvolflow + air_feed.
    nvolflow;
```

Code 4.275 calculates the volume flow at standard conditions after the mixing of the fuel gas and the combustion air streams.

Code 4.276: Elementary mass balances

```
fAr: exhaust.massflow *  exhaust.Gas.wAr = air_feed.massflow *
    air_feed.Gas.wAr + gas_feed.massflow * gas_feed.Gas.wAr;
if1 ref(gas_feed.Tar) then
f1C: exhaust.massflow * 12.011 *(2*exhaust.Gas.wC2H4/28.0536 + 2*
    exhaust.Gas.wC2H6/30.0694 + 3*exhaust.Gas.wC3H8/44.0962 +
    exhaust.Gas.wCH4/16.0428 + exhaust.Gas.wCO/28.0104 + exhaust.
    Gas.wCO2/44.0098 + exhaust.Gas.wHCN/27.02568) = massflow_mix *
    12.011 * (2*Gas_mix.wC2H4/28.0536 + 2*Gas_mix.wC2H6/30.0694 +
    3*Gas_mix.wC3H8/44.0962 + Gas_mix.wCH4/16.0428 + Gas_mix.wCO
    /28.0104 + Gas_mix.wCO2/44.0098 + Gas_mix.wHCN/27.02568) + (
    gas_feed.nvolflow/1000.0)*(gas_feed.tar_content*gas_feed.Tar.wC
    );
                              .
                              .
f1Cl: exhaust.massflow * 35.4527 * exhaust.Gas.wHCl/36.46064 =
    massflow_mix * 35.4527 * Gas_mix.wHCl/36.46064 + (gas_feed.
    nvolflow/1000.0)*(gas_feed.tar_content*gas_feed.Tar.wCl);
else1
f2C: exhaust.massflow * 12.011 *(2*exhaust.Gas.wC2H4/28.0536 + 2*
    exhaust.Gas.wC2H6/30.0694 + 3*exhaust.Gas.wC3H8/44.0962 +
    exhaust.Gas.wCH4/16.0428 + exhaust.Gas.wCO/28.0104 + exhaust.
    Gas.wCO2/44.0098 + exhaust.Gas.wHCN/27.02568) = massflow_mix *
    12.011 * (2*Gas_mix.wC2H4/28.0536 + 2*Gas_mix.wC2H6/30.0694 +
    3*Gas_mix.wC3H8/44.0962 + Gas_mix.wCH4/16.0428 + Gas_mix.wCO
    /28.0104 + Gas_mix.wCO2/44.0098 + Gas_mix.wHCN/27.02568);
```

```
                                .
                                .
f2Cl:    exhaust.massflow * 35.4527 * exhaust.Gas.wHCl/36.46064 =
    massflow_mix * 35.4527 * Gas_mix.wHCl/36.46064;
endifl
```

The definition of the exhaust gas composition starts with the elementary mass balance for C, H, O, N, S and Cl in Code 4.276 and continues with all the upcoming codes of this unit. It is distinguished between the case of tar presents in the fuel gas stream or not.

Code 4.277: Hydrocrabon content for exhaust gas stream

```
frC2H4: exhaust.Gas.wC2H4*exhaust.massflow = 0.01*HC_slip*
    massflow_mix*Gas_mix.wC2H4;
frC2H6: exhaust.Gas.wC2H6*exhaust.massflow = 0.01*HC_slip*
    massflow_mix*Gas_mix.wC2H6;
frC3H8: exhaust.Gas.wC3H8*exhaust.massflow = 0.01*HC_slip*
    massflow_mix*Gas_mix.wC3H8;
frCH4:   exhaust.Gas.wCH4*exhaust.massflow = 0.01*HC_slip*
    massflow_mix*Gas_mix.wCH4;
```

Code 4.277 defines the hydrocarbon content of the exhaust gas in interaction with the hydro carbon slip *HC_slip*-item.

Code 4.278: No hydrogen, hydrogen sulfide, hydrogen cyanide, nitrous oxide or ammonia in exhaust gas

```
frC2H4: exhaust.Gas.wC2H4*exhaust.massflow = 0.01*HC_slip*
    massflow_mix*Gas_mix.wC2H4;
frC2H6: exhaust.Gas.wC2H6*exhaust.massflow = 0.01*HC_slip*
    massflow_mix*Gas_mix.wC2H6;
frC3H8: exhaust.Gas.wC3H8*exhaust.massflow = 0.01*HC_slip*
    massflow_mix*Gas_mix.wC3H8;
frCH4:   exhaust.Gas.wCH4*exhaust.massflow = 0.01*HC_slip*
    massflow_mix*Gas_mix.wCH4;
```

The assumption has been made that neither hydrogen, hydrogen sulfide, hydrogen cyanide nor ammonia are present in the exhaust gas (Code 4.278).

Code 4.279: CO slip through the gas engine

```
frCO:    exhaust.Gas.wCO*exhaust.massflow = 0.01*CO_slip*
    massflow_mix*Gas_mix.wCO;
```

The CO slip should consider the physical slip as well as the CO slip due to incomplete combustion (Code 4.279).

Code 4.280: Switch for the thermal NOx formation in the gas engine

```
ifl  NOx_formation_model == OFF then
frNO_OFF: exhaust.massflow*exhaust.Gas.wN2 = air_feed.massflow*
    air_feed.Gas.wN2 + gas_feed.massflow*gas_feed.Gas.wN2;
endifl
ifl  NOx_formation_model == ON then
```

```
frNO_ON: exhaust.nvolflow*exhaust.Gas.yNO = nvolflow_mix*(Gas_mix.
    yHCN + 2*Gas_mix.yN2O + Gas_mix.yNH3 + Gas_mix.yNO) + exhaust.
    nvolflow*2.0*lambda^16.0*exp(-10.0*lambda);
endifl
```

Code 4.280 applies the *NOx_formation_model* switch item of Table 4.35 to the simulation. If the NOx formation switch is turned off, no thermal NO is formed and N_2 passed without reacting. If it is turned on thermal NO is formed as a function of lambda. Figure 4.42 is an

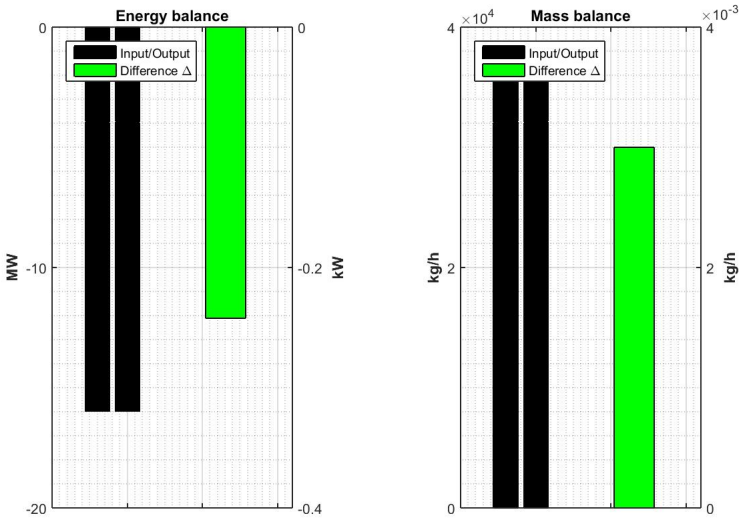


Figure 4.42: Energy and mass balance for the gas engine model

illustration of the unit's energy and mass balance.

DFB gasifier u_gasif_dfb_

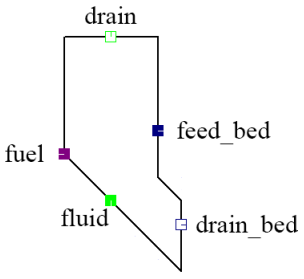


Figure 4.43: DFB gasifier library icon

There are more than this gasifier model available in the BG_Lib-library. All of them are based on or similar to this model. This gasifier model is the one applied for the dual fluidised bed gasification (DFB) simulation, see Section 3.1. The *fuel* connection supplies biomass or other organic feedstock to the gasfier. Air, steam as well as other fluidization

agents are fed via the *fluid* connection. The bed material connection *feed_bed* is compulsory to connect as well as all the other connections otherwise an error message will occur. The bed material is leaving the gasification reactor through the exit named *drain_bed*. The flue gas is removed via the *drain* outlet. The combination (DFB) of the gasifier model and a combustion chamber (riser) model is described in detail by [76][5.2].
M. Stidl [102][3.8.2] presents a gasfier model with an explanation of all the values to be set and those which are calculated, when using the model in PSE. Some of the characteristic values of this description are different to them of this model. However, this should not cause any confusion as just different parameters describe the process. All of these parameters are calculated through settings that have to be done equal for this gasifier model.

Variables	Description
dp_fluid	Pressure drop from fluidisation to producer gas drain [bar]
dp_fuel	Pressure drop from fuel to producer gas drain [bar]
dt_bed_drain	Temperature difference between producer gas and bed material drain stream [bar]
ratio_fluid_fuel	Mass of fluidisation agent per mass unit of dry fuel (organic + ash) [kg/kg]
ratio_steam_fuel	Steam to fuel ratio. This value relates the total mass of H_2O entering the gasifier to the mass of the dry fuel (organic + ash) [kg/kg]
ratio_fluid_fuel_waf	Steam to fuel ratio. This value relates the total mass of H_2O entering the gasifier to the mass of the dry and ash free fuel (only organic) [kg/kg]
ratio_steam_C	Steam to carbon ratio. This value relates the total mol of H_2O entering the gasifier to the mol of carbon from the fuel and fluidisation gas without CO_2 [mol/mol]
water_conv	Water conversion rate in the gasifier related to the mass of dry fuel (organic + ash) [kg/kg_dry]
ratio_bdox_fuel	Ratio between mass oxygen in the fuel and mass of dry fuel [kg/kg]
cycling_rate	Cycling rate (mass flow) of the bed material in relation to the mass flow of dry fuel (organic + ash) [kg/kg]
fract_tar_pg	Mass fraction of the ash free and water free fuel exiting the gasifier as tar in the producer gas [kg/kg]
fract_char_pg	Mass fraction of the fuel exiting the gasifier as ungasified char with the producer gas (fly char) [kg/kg]
fract_char_bed	Mass fraction of ash free and water free fuel exiting the gasifier as ungasified char with the bed material [kg/kg]
char_out	Total char discharge with producer gas and bed material [kg/h]
phi_gas_incl_tar	Fuel (waf) which transform to gaseous aggregate state [kg/kg_waf]
ungsfd_C	Ungasified carbon in total char related to carbon in fuel [%]
fract_fly_ash	Mass fraction of ash introduced with the fuel, which is exiting the gasifier as dust in the producer gas [kg/kg]
bed_attrition	Mass flow of bed material exiting the gasifier as dust in producer gas [kg/h]

P_therm	Thermal power of the fuel entering the gasifier based on lower heating value [kW]
Heat_gasif	Heat required for gasification of the fuel [kW]
q_loss_rel	Relative heat loss of the gasifier in % of thermal fuel power [%]
Q_loss	Heat loss of the gasifier dependent on thermal power (Power_therm). The dependency is determined by f_heat_loss [kW]
E_loss	Exergy loss [kW]

Table 4.36: Gasifier model item description

Code 4.281: Energy balance

```
fEnergy:        drain.massflow*drain.h_total + drain_bed.massflow*
    drain_bed.h_total + Q_loss*3600.0 = fuel.massflow*fuel.h_total
    + fluid.massflow*fluid.h_total + feed_bed.massflow*feed_bed.
    h_total;
```

Code 4.282: Thermal fuel power

```
fP_therm:       P_therm*3600.0 = fuel.massflow*fuel.lhv;
```

Code 4.283: Heat loss

```
fQ_loss:        Q_loss = P_therm*q_loss_rel/100.0;
```

Code 4.281 shows the energy balance of the gasifier. The thermal power equals the fuel energy, based on the lower heating value (Code 4.282). The overall heat loss is calculated in relation to the relative heat loss and the thermal power (Code 4.283).

Code 4.284: Exergy loss

```
fE_loss: drain.Exergy + drain_bed.Exergy + E_loss =
        fuel.Exergy + fluid.Exergy + feed_bed.Exergy;
```

Through the balance of Code 4.284 the exergy loss is calculated.

Code 4.285: Pressure drops

```
fdp_fluid:      fluid.p = drain.p + dp_fluid;
fdp_fuel:       fuel.p = drain.p + dp_fuel;
```

Code 4.286: Pressure drops

```
fdt_bed_drain:  drain_bed.t = drain.t + dt_bed_drain;
```

Pressure drops and the temperature difference between the gasifier product gas and the bed material drain stream are obtained via Code 4.285 and Code 4.286.

Code 4.287: Elementary mass balances

```
if!ref(feed_bed.Gas) then
f1Ar:    drain.massflow*drain.Gas.wAr = fluid.massflow*fluid.Gas.
    wAr;
f1C:     drain.massflow*12.011*(2*drain.Gas.wC2H4/28.0536 + 2*drain
    .Gas.wC2H6/30.0694 + 3*drain.Gas.wC3H8/44.0962 + drain.Gas.wCH4
    /16.0428 + drain.Gas.wCO/28.0104 + drain.Gas.wCO2/44.0098 +
    drain.Gas.wHCN/27.02568) + drain.nvolflow*((drain.char_content
    /1000.0)*drain.Char.wC + (drain.tar_content/1000.0)*drain.Tar.
    wC) + drain_bed.massflow*drain_bed.char_content*drain_bed.Char.
    wC = fuel.massflow*(1.0 - fuel.water_content - fuel.ash_content
    )*fuel.Organic.wC + fluid.massflow*12.011*(2*fluid.Gas.wC2H4
    /28.0536 + 2*fluid.Gas.wC2H6/30.0694 + 3*fluid.Gas.wC3H8
    /44.0962 + fluid.Gas.wCH4/16.0428 + fluid.Gas.wCO/28.0104 +
    fluid.Gas.wCO2/44.0098 + fluid.Gas.wHCN/27.02568);
                              .
f1Cl:    drain.massflow*35.4527*drain.Gas.wHCl/36.46064 + drain.
    nvolflow*((drain.char_content/1000.0)*drain.Char.wCl + (drain.
    tar_content/1000.0)*drain.Tar.wCl) + drain_bed.massflow*
    drain_bed.char_content*drain_bed.Char.wCl = fuel.massflow*(1.0
    - fuel.water_content - fuel.ash_content)*fuel.Organic.wCl +
    fluid.massflow*35.4527*fluid.Gas.wHCl/36.46064;
elsel
f2Ar:    drain.massflow*drain.Gas.wAr = fluid.massflow*fluid.Gas.
    wAr + feed_bed.massflow*feed_bed.gas_ads*0.001*feed_bed.Gas.M*
    feed_bed.Gas.wAr;
f2C:     drain.massflow*12.011*(2*drain.Gas.wC2H4/28.0536 + 2*drain
    .Gas.wC2H6/30.0694 + 3*drain.Gas.wC3H8/44.0962 + drain.Gas.wCH4
    /16.0428 + drain.Gas.wCO/28.0104 + drain.Gas.wCO2/44.0098 +
    drain.Gas.wHCN/27.02568) + drain.nvolflow*((drain.char_content
    /1000.0)*drain.Char.wC + (drain.tar_content/1000.0)*drain.Tar.
    wC) + drain_bed.massflow*drain_bed.char_content*drain_bed.Char.
    wC = fuel.massflow*(1.0 - fuel.water_content - fuel.ash_content
    )*fuel.Organic.wC + fluid.massflow*12.011*(2*fluid.Gas.wC2H4
    /28.0536 + 2*fluid.Gas.wC2H6/30.0694 + 3*fluid.Gas.wC3H8
    /44.0962 + fluid.Gas.wCH4/16.0428 + fluid.Gas.wCO/28.0104 +
    fluid.Gas.wCO2/44.0098 + fluid.Gas.wHCN/27.02568) + feed_bed.
    massflow*feed_bed.gas_ads*0.001*feed_bed.Gas.M*12.011*(2*
    feed_bed.Gas.wC2H4/28.0536 + 2*feed_bed.Gas.wC2H6/30.0694 + 3*
    feed_bed.Gas.wC3H8/44.0962 + feed_bed.Gas.wCH4/16.0428 +
    feed_bed.Gas.wCO/28.0104 + feed_bed.Gas.wCO2/44.0098 + feed_bed
    .Gas.wHCN/27.02568);
                              .
f2Cl:    drain.massflow*35.4527*drain.Gas.wHCl/36.46064 + drain.
    nvolflow*((drain.char_content/1000.0)*drain.Char.wCl + (drain.
    tar_content/1000.0)*drain.Tar.wCl) + drain_bed.massflow*
    drain_bed.char_content*drain_bed.Char.wCl = fuel.massflow*(1.0
    - fuel.water_content - fuel.ash_content)*fuel.Organic.wCl +
    fluid.massflow*35.4527*fluid.Gas.wHCl/36.46064 + feed_bed.
    massflow*feed_bed.gas_ads*0.001*feed_bed.Gas.M*35.4527*feed_bed
    .Gas.wHCl/36.46064;
endifl
```

Code 4.287 is an elementary mass balance of the overall gasifier. For its first case, it is assumed that no gas is absorbed in the entering bed material. The second cases considers the vice versa scenario. The two cases are separated by the *elsel* command.
Gaseous plus organic phase are treated separately from the inorganic phase. Inorganic components are considered as chemically invariant. This model assumption stands in contrast to the gasifier model of SER-gasification. For the SER-model most parts are equal to this model, except the elementary balance where inorganic components are considered.
In addition to the balances of Code 4.287 eleven setting values in the drain gas composition are required to fully define the gasifier model. All exiting organic globals (char, tar) are strictly defined by setting values as well. All of this last paragraph's information is well illustrated by M. Stidl [102][p.89].

Code 4.288: Inorganic species balances

```
fAsh:    drain.nvolflow*(drain.dust_content/1000.0)*drain.Dust.wAsh
     + drain_bed.massflow*drain_bed.Solid.wAsh = fuel.massflow*fuel
    .ash_content*fuel.Ash.wAsh + feed_bed.massflow*feed_bed.Solid.
    wAsh;
fK2O:    drain.nvolflow*(drain.dust_content/1000.0)*drain.Dust.wK2O
     + drain_bed.massflow*drain_bed.Solid.wK2O = fuel.massflow*fuel
    .ash_content*fuel.Ash.wK2O + feed_bed.massflow*feed_bed.Solid.
    wK2O;
                                        .
                                        .
fCaOH2: drain.nvolflow*(drain.dust_content/1000.0)*drain.Dust.
    wCaOH2 + drain_bed.massflow*drain_bed.Solid.wCaOH2 = fuel.
    massflow*fuel.ash_content*fuel.Ash.wCaOH2 + feed_bed.massflow*
    feed_bed.Solid.wCaOH2;
```

For the balance of inorganic/solid components ash, K_2O, MgO, CaO, SiO_2, $CaCO_3$, dolomite, $CaSO_4$ and $CaOH_2$ are considered, see Code 4.287.

Code 4.289: Organic substances splitting

```
ffract_char_bed: drain_bed.massflow*drain_bed.char_content = fuel.
    massflow*(1.0 - fuel.water_content - fuel.ash_content)*
    fract_char_bed;
ffract_char_pg: drain.nvolflow*(drain.char_content/1000.0) = fuel.
    massflow*(1.0 - fuel.water_content - fuel.ash_content)*
    fract_char_pg;
ffract_tar_pg: drain.nvolflow*(drain.tar_content/1000.0) = fuel.
    massflow*(1.0 - fuel.water_content - fuel.ash_content)*
    fract_tar_pg;
fchar_out: char_out = drain.nvolflow*(drain.char_content/1000.0) +
     drain_bed.massflow*drain_bed.char_content;
fphi_gas_incl_tar:       char_out = fuel.massflow*(1.0 - fuel.
    water_content - fuel.ash_content) * (1 - phi_gas_incl_tar);
fungsfd_C: drain_bed.massflow*drain_bed.char_content*drain_bed.
    Char.wC + drain.nvolflow*(drain.char_content/1000.0)*drain.Char
    .wC = fuel.massflow*(1.0 - fuel.water_content - fuel.
    ash_content)*fuel.Organic.wC*ungsfd_C/100.0;
```

Characteristic values to described the splitting of organic substances are calculated by Code 4.289.

Code 4.290: Inorganic substances splitting

```
fdust_Ash: drain.nvolflow*(drain.dust_content/1000.0)*drain.Dust.
    wAsh = fuel.massflow*fuel.ash_content*fract_fly_ash*fuel.Ash.
    wAsh      + bed_attrition*feed_bed.Solid.wAsh;
fdust_K2O: drain.nvolflow*(drain.dust_content/1000.0)*drain.Dust.
    wK2O = fuel.massflow*fuel.ash_content*fract_fly_ash*fuel.Ash.
    wK2O      + bed_attrition*feed_bed.Solid.wK2O;
                          .
                          .
fdust_CaOH2: drain.nvolflow*(drain.dust_content/1000.0)*drain.Dust
    .wCaOH2 = fuel.massflow*fuel.ash_content*fract_fly_ash*fuel.Ash
    .wCaOH2 + bed_attrition*feed_bed.Solid.wCaOH2;
```

The set of equations for inorganic components (Code 4.288/4.290) are responsible for the definition of the inorganic composition of the drain gas and drain bed material stream as well as the mass flow of the exiting bed material stream.

Code 4.291: Fluidisation/fuel ratio

```
fratio_fluid_fuel:      ratio_fluid_fuel*(fuel.massflow*(1 - fuel.
    water_content)) = fluid.massflow;
```

Code 4.292: Steam/fuel wf ratio

```
fratio_steam_fuel:      ratio_steam_fuel*(fuel.massflow*(1 - fuel.
    water_content)) = fluid.massflow*fluid.Gas.wH2O + fuel.massflow
    *fuel.water_content;
```

Code 4.293: Steam/fuel waf ratio

```
fratio_steam_fuel_waf: ratio_steam_fuel_waf*(fuel.massflow*(1 -
    fuel.water_content - fuel.ash_content)) = fluid.massflow*fluid.
    Gas.wH2O + fuel.massflow*fuel.water_content;
```

Code 4.294: Steam/carbon ratio

```
fratio_steam_C: ratio_steam_C * (fuel.massflow*(1 - fuel.
    water_content - fuel.ash_content)*fuel.Organic.wC/12.011 +
    fluid.massflow*(2*fluid.Gas.wC2H4/28.0536 + 2*fluid.Gas.wC2H6
    /30.0694 + 3*fluid.Gas.wC3H8/44.0962 + fluid.Gas.wCH4/16.0428 +
     fluid.Gas.wCO/28.0104 + fluid.Gas.wHCN/27.02568))= (fluid.
    massflow*fluid.Gas.wH2O + fuel.massflow*fuel.water_content)
    /18.01528;
```

Code 4.295: Ratio mass additional oxygen/mass dry fuel

```
ifl ref(feed_bed.Gas) then
f1ratio_bdox_fuel: (1 - fuel.water_content)*fuel.massflow*
    ratio_bdox_fuel = feed_bed.massflow*feed_bed.gas_ads*feed_bed.
    Gas.yO2*31.9988/1000.0;
elsel
f2ratio_bdox_fuel: ratio_bdox_fuel = 0.0;
endifl
```

Code 4.296: Cycling rate

```
fcycling_rate:  cycling_rate*fuel.massflow*(1 - fuel.water_content
    ) = drain_bed.massflow*(1.0 + drain_bed.char_content);
```

Code 4.297: Heat from circulating bed material

```
fHeat_gasif:  Heat_gasif*3600.0 = feed_bed.massflow*feed_bed.h -
    drain_bed.massflow*drain_bed.h;
```

All the generated characteristic values -Code 4.291 to 4.297- are obtained in order to describe the operation conditions of the gasifier. The fluidization to fuel ratio (Code 4.291) is defined as mass of fluidization agent per mass of dry fuel (organic + inorganic). The steam to fuel ratio is defined as total mass of water per mass of dry (wf) fuel (organic + inorganic), Code 4.292. The second steam to fuel ratio is described as total mass of water per mass of dry ash free (waf) fuel (only organic), Code 4.293. The steam to carbon ratio is calculated as total mass of water per mass of all carbon present (fuel carbon + carbon in fluidization medium, without CO_2), Code 4.294. The cycling rate is defined as mass of inorganic bed material divided through the mass of dry fuel (organic + inorganic), Code 4.296. The other values are self-explanatory and do not need any further comments.

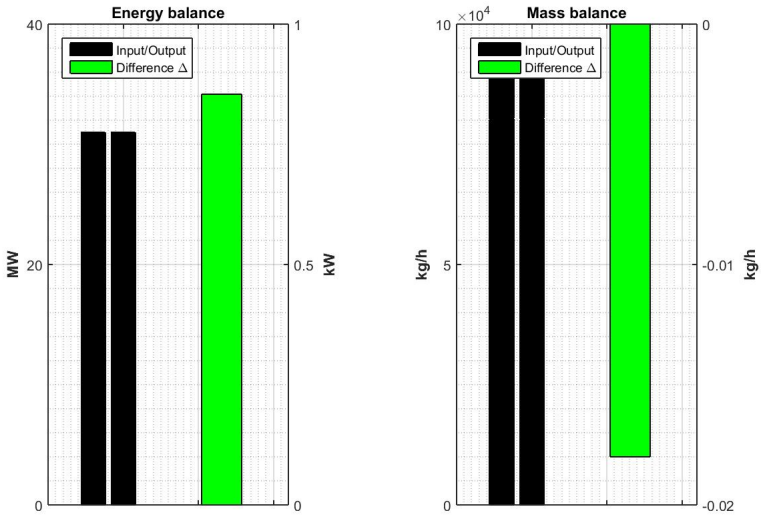


Figure 4.44: Energy and mass balance for the DFB-gasifier unit

Figure 4.44 shows the energy and mass balance of this unit.

Generator u_gen_el_

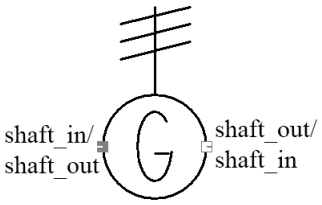


Figure 4.45: Generator library icon

As described for several units before, also these shaft connectors are defined as in- and outlet at the same time, due to the rotary mode.

Variables	Description
power	Electrical power output [kW]
eta_el	Electrical efficiency [%]
eta_m	Mechanical efficiency [%]
E_loss	Exergy loss [kW]

Table 4.37: Generator model item description

Code 4.298: Energy balance

```
ifl ref(shaft_in) && !ref(shaft_out) then
        f1a:    power - (eta_el/100)*(eta_m/100)*shaft_in.power =
            0.0;
        f_E_loss_a:     shaft_in.power = power + E_loss;
endifl

# shaft only from right
ifl !ref(shaft_in) && ref(shaft_out) then
        f1b:    power + (eta_el/100)*(eta_m/100)*shaft_out.power =
             0.0;
        f_E_loss_b:     0=shaft_out.power+power+E_loss;
endifl

# shaft from both sides
ifl ref(shaft_in) && ref(shaft_out) then
        f1c:    power + (eta_el/100)*(eta_m/100)*(shaft_out.power-
            shaft_in.power) = 0.0;
        f_E_loss_c:     shaft_in.power=shaft_out.power+power+
            E_loss;
endifl
```

The energy balance (Code 4.298) at first considers the case when mechanical energy is fed to the generator from the left side. The second case shows an mechanical output at the right side, the left connector is not used. Case three considers supply from the left side and

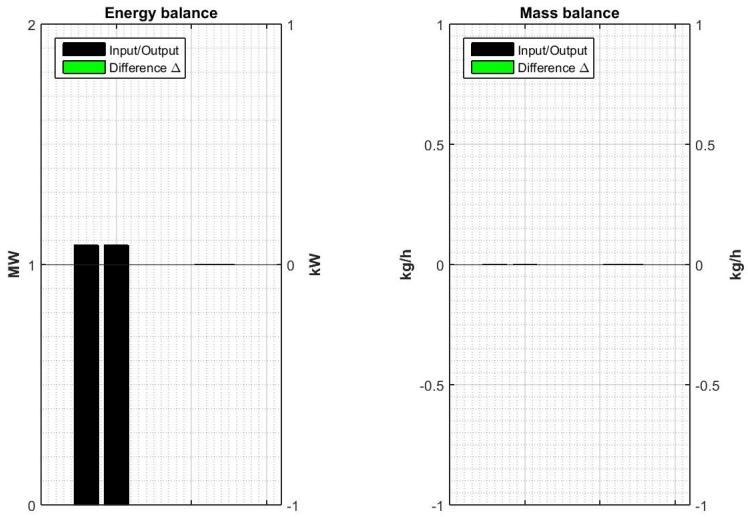


Figure 4.46: Energy and mass balance for the generator unit

output to the right side. This unit does not consider any mass transport. Mechanical energy is converted into electricity, therefore only an energy balance can be carried out, see Figure 4.46.

Heat exchanger gas-gas u_htex_gg_

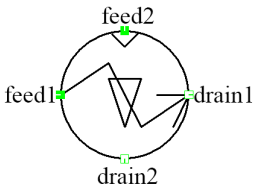


Figure 4.47: Heat exchanger library icon

Heat transport from one to another stream is maintained by an heat exchanger unit. This explanation for heat exchange between two gaseous streams is exemplary for any other heat exchanger model. Each and every source code is equal despite the fact that e.g. for heat exchange between an organic and gaseous stream the *feed1* connector is for organic streams. But still, even the names of the connectors are equal. Therefore, just the heat exchanger unit for gas-gas heat exchange is described. One stream enters at the inlet *feed1* and leaves at *drain1*. A second one is fed via *feed2* and leaves at *drain2*. Additional theoretical information can be found in [76][5.1.3].

Switch	Description
Type	Defines heat exchanger type, it can be chosen between Co- and Counter-current
Variables	**Description**
dp_1	Pressure drop of stream 1, which is connected by the zick-zack arrow [bar]
dp_2	Pressure drop of stream 2, which is not connected by the zick-zack arrow in the symbol [bar]
dt_in_1	Temperature difference between the two streams at the entry side of stream 1, which is connected by the zick-zack arrow [°C]
dt_out_1	Temperature difference between the two streams at the exit side of stream 1, which is connected by the zick-zack arrow [°C]
kA	Heat transfer coefficient multiplied by transfer area [kW/K]
Q_trans	Transferred heat [kW]
E_loss	Exergy loss [kW]

Table 4.38: Heat exchanger model item description

Code 4.299: Mass balance

```
fmass1: feed1.massflow = drain1.massflow;
fmass2: feed2.massflow  = drain2.massflow;
```

Code 4.300: Pressure drop

```
fdp1:   feed1.p - dp_1 = drain1.p;
fdp2:   feed2.p - dp_2 = drain2.p;
```

Mass balances and pressure drops for both streams are defined separately by Code 4.299 and 4.300.

Code 4.301: Energy balance

```
fh1:    if (feed1.t < feed2.t)  then feed1.massflow*feed1.h_total
   + Q_trans*3600.0 = drain1.massflow*drain1.h_total;
else feed1.massflow*feed1.h_total - Q_trans*3600.0 = drain1.
   massflow*drain1.h_total;
fh2:    if (feed1.t < feed2.t)  then feed2.massflow*feed2.h_total
   = drain2.massflow*drain2.h_total + Q_trans*3600.0;
else feed2.massflow*feed2.h_total = drain2.massflow*drain2.h_total
    - Q_trans*3600.0;
```

The choice of energy balance in Code 4.301 depends on the temperature level of *feed1* and *feed2*, or to put it another way, which one is higher respectively lower.

Code 4.302: Temperature difference

```
if1 Type == CoCurrent then
fdt_co1:        if (feed1.t < feed2.t)  then dt_in_1 = feed2.t -
   feed1.t;
                        else dt_in_1 = feed1.t - feed2.t;
fdt_co2:        if (feed1.t < feed2.t)  then dt_out_1 = drain2.t -
    drain1.t;
                        else dt_out_1 = drain1.t - drain2.t;
endif1
if1 Type == CounterCurrent then
fdt_cc1:        if (feed1.t < feed2.t)  then dt_in_1 = drain2.t -
   feed1.t;
                        else dt_in_1 = drain1.t - feed2.t;
fdt_cc2:        if (feed1.t < feed2.t)  then dt_out_1 = feed2.t -
   drain1.t;
                        else dt_out_1 = feed1.t - drain2.t;
endif1
```

The various temperature differences of Code 4.302 are calculated for Co- and Countercurrent operation of the heat exchanger. The temperature levels of the streams *feed1* and *feed2* are a factor for the correct equation selection as well.

Code 4.303: Temperature difference

```
fdelta_t_eff:   if (abs(dt_in_1/dt_out_1) >= 1.2 || abs(dt_out_1/
   dt_in_1) >= 1.2) && (dt_in_1 > 0) && (dt_out_1 > 0)
then
        Q_trans*ln(dt_in_1/dt_out_1)/(dt_in_1 - dt_out_1) = kA;
else
        Q_trans = kA*0.5*(dt_in_1 + dt_out_1);
```

If the if-conditions of Code 4.303 are met, the logarithmic average temperature difference is calculated according to equation 4.10. *dt_in_1* equals the larger, maximumm temperature difference dt_{max} and *dt_out_1* the smaller one.

$$dt_{m,log} = \frac{(dt_{max} - dt_{min})}{ln\frac{dt_{max}}{dt_{min}}} \tag{4.10}$$

If the conditions from the first line of Code 4.303 are not met, the sum of both temperature differences is simply halved. *kA* is the quantifying value for a heat exchanger. A [m^2] describes the exchange area and k [W/m^2K] the heat transition coefficient.

Code 4.304: Exergy loss

```
f_E_loss: E_loss = feed1.Exergy + feed2.Exergy - drain1.Exergy -
    drain2.Exergy;
```

The exergy loss is calculated by the balance of Code 4.304.

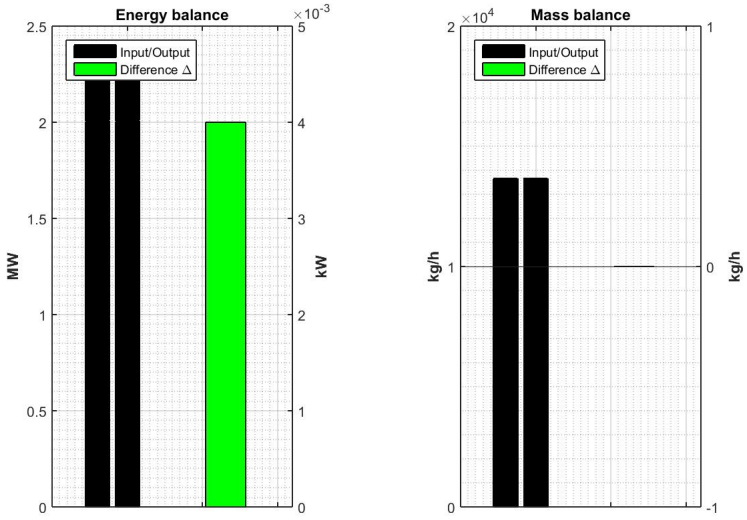


Figure 4.48: Energy and mass balance for the heat exchanger unit

Figure 4.48 is the illustration of the heat exchanger's energy balance.

Hydraulic switch u_hydraulic_switch_

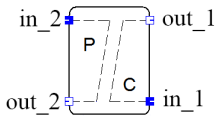


Figure 4.49: Hydraulic switch library icon

The hydraulic switch unit is for the simulation of district heating/cooling approaches. A water stream, carrying waste energy, is fed to the unit via connector *in_2*. The usable share of energy is brought to the consumer side and the residual energy remains at the producer side and leaves via exit *out_2*. On the consumer side the energy carrier is also water. It enters at *in_1* and leaves -at a higher energy level- at outlet *out_1*.

Variables	Description
Q_Cons	Exchanged energy [kW]
dp_1	Pressure drop at consumer side [bar]
dp_2	Pressure drop at producer side [bar]
E_loss	Exergy loss [kW]

Table 4.39: Hydraulic switch model item description

Code 4.305: Mass balance

```
fmass: in_2.massflow + in_1.massflow = out_1.massflow + out_2.
    massflow;
```

Code 4.306: Energy balance

```
fh_total: in_1.h_total*in_1.massflow + in_2.h_total*in_2.massflow
    = out_1.h_total*out_1.massflow + out_2.massflow*out_2.h_total;
```

Code 4.307: Exchanged energy

```
fQ_Cons: Q_Cons = (in_2.massflow *(in_2.h - in_2.wfhpt
    (1.00000,25.0)) / 3600) -  (out_2.massflow*(out_2.h - out_2.
    wfhpt(1.00000,25.0))/3600);
```

The unit's overall mass and energy balance are shown in Code 4.305 and 4.306. The exchanged energy is calculated by the enthalpies, they are based on 25°C, see Code 4.307.

Code 4.308: Pressure drops

```
fp1:      in_1.p = out_1.p + dp_1;
fp2:      in_2.p = out_2.p + dp_2;
```

Code 4.309: Exergy loss

```
f_E_loss: in_1.Exergy + in_2.Exergy = out_1.Exergy + out_2.Exergy
    + E_loss;
```

The pressure drops on consumer and producer side are calculated by Code 4.308 and the exergy loss by Code 4.309. Figure 4.50 presents the unit's energy and mass balance.

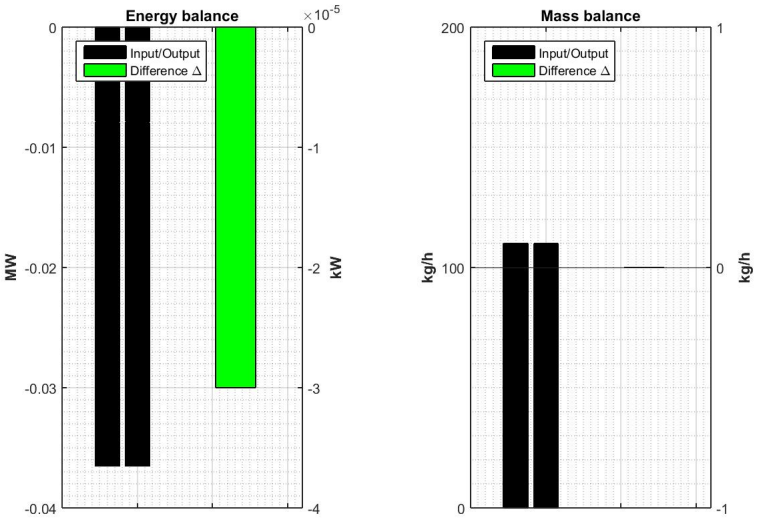


Figure 4.50: Energy and mass balance for the hydraulic switch

Info unit gas u_info_g_

Figure 4.51: Info unit library icon

The Info-unit was established to provid detailed information on energy and mass flows for the different stream classes. Exemplary the info-unit for gaseous streams is described in the following subsection. Further information on its advantages for the creation of energy balances can be found in the dissertation of M. Stidl [102].
As presented in Figure 4.51 the gas stream enters at the connector called *feed* and leaves at the *drain* connector.

Variables	Description
massflow_gas	Massflow gas (without dust, char and tar) [kg/h]
massflow_dust	Mass flow dust only [kg/h]
massflow_char	Mass flow char only [kg/h]
massflow_tar	Mass flow tar only [kg/h]
Mass_total	Mass flow of whole gas stream (including dust, char and tar) [kg_total/h]
Q_298_gas	Sensible heat (enthalpy) of gas stream (sensible heat = 0 at 25°C (298.15 K)) [kW]
Q_298_dust	Sensible heat (enthalpy) of dust content (sensible heat = 0 at 25°C (298.15 K)) [kW]
Q_298_char	Sensible heat (enthalpy) of char content (sensible heat = 0 at 25°C (298.15 K)) [kW]
Q_298_tar	Sensible heat (enthalpy) of tar content (sensible heat = 0 at 25°C (298.15 K)) [kW]
Q_298_total	Sensible heat (enthalpy) of total stream (sensible heat = 0 at 25°C (298.15 K)) [kW]
LHV_gas	Lower heating value specific to gas mass at 25°C (298.15 K) [kJ/kg_gas]
LHV_char	Lower heating value char content specific to char mass at 25°C (298.15 K) [kJ/kg_char]
LHV_tar	Lower heating value tar content specific to tar mass at 25°C (298.15 K) [kJ/kg_tar]
LHV_total	Lower heating value specific to total mass at 25°C (298.15 K) [kJ/kg_total]
P_chem_gas	Chemical power gas (without char and tar) based to the lower heating value (LHV evaluating temperature: 25°C) [kW]
P_chem_char	Chemical power char content based to the lower heating value (LHV evaluating temperature: 25°C) [kW]
P_chem_tar	Chemical power tar content based to the lower heating value (LHV evaluating temperature: 25°C) [kW]
P_chem_total	Chemical power total stream based to the lower heating value (LHV evaluating temperature: 25°C) [kW]

Variables	Description
Q_298_and_P	Sum of sensible heat and chemical power (based on lower heating value) of the whole gas stream (including dust, char and tar). Sensible heat and lower heating value are 0 at 25°C (298.15 K) [kW]
HHV_gas	Higher heating value specific to gas mass at 25°C (298.15 K) [kJ/kg_gas]
HHV_char	Higher heating value of char content specific to char mass at 25°C (298.15 K) [kJ/kg_char]
HHV_tar	Higher heating value of tar content specific to tar mass at 25°C (298.15 K) [kJ/kg_tar]
HHV_total	Higher heating value specific to total mass at 25°C (298.15 K) [kJ/kg_total]
P_ch_HHV_gas	Chemical power gas (without char and tar) based to the higher heating value (HHV evaluating temperatur: 25°C) [kW]
P_ch_HHV_char	Chemical power char content based to the higher heating value (HHV evaluating temperatur: 25°C) [kW]
P_ch_HHV_tar	Chemical power tar content based to the higher heating value (HHV evaluating temperatur: 25°C) [kW]
P_ch_HHV_total	Total chemical power based to the higher heating value (HHV evaluating temperatur: 25°C) [kW]
Q_298_and_P_HHV	Sum of sensible heat and chemical power (based on higher heating value) of the whole gas stream (including dust, char and tar). Sensible heat and higher heating value are 0 at 25°C (298.15 K) [kW]
H_f_298_gas	Enthalpy of formation from the gas (without dust, char and tar) at 25°C (298.15 K). It is 0 for the thermodynamically stable modification of a pure element (for example Ar, C, H_2, N_2) at 25°C (298.15 K) and 1.00000 bar [kW]
H_f_298_dust	Enthalpy of formation from the dust content at 25°C (298.15 K). It is 0 for the thermodynamic stable modification of a pure element (for example Ar, C, H_2, N_2) at 25°C (298.15 K) and 1.00000 bar [kW]
H_f_298_char	Enthalpy of formation from the char content at 25°C (298.15 K). It is 0 for the thermodynamic stable modification of a pure element (for example Ar, C, H_2, N_2) at 25°C (298.15 K) and 1.00000 bar [kW]
H_f_298_tar	Enthalpy of formation from the tar content at 25°C (298.15 K). It is 0 for the thermodynamic stable modification of a pure element (for example Ar, C, H_2, N_2) at 25°C (298.15 K) and 1.00000 bar [kW]
H_f_298_total	Enthalpy of formation for the whole gas stream (including dust, char and tar) at 25°C (298.15 K). It is 0 for the thermodynamic stable modification of a pure element (for example Ar, C, H_2, N_2) at 25°C (298.15 K) and 1.00000 bar [kW]
H_total	Total thermodynamic enthalpy of the whole gas stream (including dust, char and tar). It will be calculated from the sum between the sensible heat and the enthalpy of formation [kW]

Table 4.40: Info model item description

Code 4.310: Equality of temperature

```
f_temp: drain.t = feed.t;
```

Code 4.311: Continuity of mass

```
f_mass: drain.massflow = feed.massflow;
ifl ref(feed.Dust) then f_dust_content: drain.dust_content = feed.
    dust_content; endifl
ifl ref(feed.Char) then f_char_content: drain.char_content = feed.
    char_content; endifl
ifl ref(feed.Tar) then  f_tar_content:  drain.tar_content = feed.
    tar_content;    endifl
```

Code 4.312: Equality of pressure

```
fQ_Cons: Q_Cons = (in_2.massflow *(in_2.h - in_2.wfhpt
    (1.00000,25.0)) / 3600) -  (out_2.massflow*(out_2.h - out_2.
    wfhpt(1.00000,25.0))/3600);
```

Code 4.310 to 4.312 are responsible for temperature, mass and pressure continuity.

Code 4.313: Mass balance

```
f_massflow_gas: massflow_gas = feed.massflow;
ifl ref(feed.Dust) then
   f_massflow_dust:      massflow_dust = feed.nvolflow*feed.
       dust_content/1000;
elsel
   f_massflow_dust2:     massflow_dust = 0;
endifl
ifl ref(feed.Char) then
   f_massflow_char:      massflow_char = feed.nvolflow*feed.
       char_content/1000;
elsel
  f_massflow_char2:      massflow_char = 0;
endifl
ifl ref(feed.Tar) then
   f_massflow_tar:       massflow_tar = feed.nvolflow*feed.
       tar_content/1000;
elsel
   f_massflow_tar2:      massflow_tar = 0;
endifl
```

In dependence of the presence of dust, char and/or tar one of the various mass balances of Code 4.313 is applied.

Code 4.314: Total mass flow

```
f_mass_total:    Mass_total = massflow_gas + massflow_dust +
    massflow_char + massflow_tar;
```

The total mass flow is defined as sum of gas, dust, char and tar share, see Code 4.314.

Code 4.315: LHV gas

```
f_lhv_gas:      LHV_gas * feed.Gas.M * 101.325 = feed.Gas.LHV *
    1000 * 8.31451 * 273.15;
```

In Code 4.315 the standard volume flow is converted into a massflow to further calculate the variable *LHV_gas* in kJ/kg_gas.

Code 4.316: LHV various contents

```
ifl ref(feed.Char) then
   f_lhv_char:  LHV_char = feed.Char.lhv;
elsel
   f_lhv_char2:          LHV_char = 0;
endifl

ifl ref(feed.Tar) then
   f_lhv_tar:            LHV_tar = feed.Tar.lhv;
elsel
   f_lhv_tar2:           LHV_tar = 0;
endifl
```

Code 4.317: LHV total

```
f_lhv_total:    LHV_total * (feed.Gas.M * 101.325 / (8.31451 *
    273.15) + feed.dust_content / 1000 + feed.char_content / 1000 +
     feed.tar_content / 1000) = LHV_gas * feed.Gas.M * 101.325 /
    (8.31451 * 273.15) + LHV_char * feed.char_content / 1000 +
    LHV_tar * feed.tar_content / 1000;
```

If char or tar are present in the stream of interest, the LHV output is generated due to the values of the *feed* stream. If not, it is set to zero (Code 4.316). The total LHV quantity *LHV_total* is calculated as sum of all the containing shares, Code 4.317.

Code 4.318: HHV gas

```
f_hhv_gas:      HHV_gas * feed.Gas.M * 101.325 = feed.Gas.HHV *
    1000 * 8.31451 * 273.15;
```

Code 4.319: HHV various contents

```
ifl ref(feed.Char) then
   f_hhv_char:  HHV_char = feed.Char.hhv;
elsel
   f_hhv_char2:          HHV_char = 0;
endifl
ifl ref(feed.Tar) then
   f_hhv_tar:            HHV_tar = feed.Tar.hhv;
elsel
   f_hhv_tar2:           HHV_tar = 0;
endifl
```

Code 4.320: HHV total

```
f_hhv_total:    HHV_total * (feed.Gas.M * 101.325 / (8.31451 *
    273.15) + feed.dust_content / 1000 + feed.char_content / 1000 +
     feed.tar_content / 1000) = HHV_gas * feed.Gas.M * 101.325 /
    (8.31451 * 273.15) + HHV_char * feed.char_content / 1000 +
    HHV_tar * feed.tar_content / 1000;
```

The calculation of the HHV-values by Code 4.318, 4.319 and 4.320 follow the same principle as described above for the LHV.

Code 4.321: Sensible heat

```
f_q_298_gas:    Q_298_gas = (feed.h - feed.Gas.gfht(25)) *
    massflow_gas / 3600;
if1 ref(feed.Dust) then
   f_q_298_dust:        Q_298_dust = (feed.h_dust - feed.Dust.sfht
       (25)) * massflow_dust / 3600;
else1
   f_q_298_dust2:       Q_298_dust = 0;
endif1
if1 ref(feed.Char) then
   f_q_298_char:        Q_298_char = (feed.h_char - feed.Char.ofht
       (25.0)) * massflow_char / 3600;
else1
   f_q_298_char2:       Q_298_char = 0;
endif1
if1 ref(feed.Tar) then
   f_q_298_tar:         Q_298_tar = (feed.h_tar - feed.Tar.ofht
       (25.0)) * massflow_tar / 3600;
else1
   f_q_298_tar2:        Q_298_tar = 0;
endif1

f_q_298_total:  Q_298_total = Q_298_gas + Q_298_dust + Q_298_char
   + Q_298_tar;
```

The sensible heat of the various stream components is generally defined as the specific heat capacity (cp) times temperature difference (dT). In this case the temperature difference is from the actual temperature to 25°C and $cp \cdot dT$ is expressed by the enthalpy difference. If one of the components dust, char or tar is not present its sensible enthalpy values is set to zero as for LHV and HHV above. The sum of all sensible heat is defined as *Q_298_total*.

Code 4.322: Sensible heat and chemical power

```
f_q_298_and_p:  Q_298_and_P = Q_298_total + P_chem_total;
f_q_298_and_p_hhv:      Q_298_and_P_HHV = Q_298_total +
    P_ch_HHV_total;
```

The chemical power based on the LHV and HHV is calculated below (Code 4.323 and 4.324) and summed up with the sensible heat in Code 4.322 to define the variables Q_298_and_P and Q_298_and_P_HHV.

Code 4.323: Chemical power based on lower heating value

```
f_p_chem_gas:    P_chem_gas = LHV_gas * massflow_gas / 3600;
f_p_chem_char:   P_chem_char = LHV_char * massflow_char / 3600;
f_p_chem_tar:    P_chem_tar = LHV_tar * massflow_tar / 3600;
f_p_chem_total: P_chem_total = P_chem_gas + P_chem_char +
    P_chem_tar;
```

Code 4.324: Chemical power based on higher heating value

```
f_p_ch_hhv_gas: P_ch_HHV_gas = HHV_gas * massflow_gas / 3600;
f_p_ch_hhv_char:        P_ch_HHV_char = HHV_char * massflow_char /
     3600;
f_p_ch_hhv_tar: P_ch_HHV_tar = HHV_tar * massflow_tar / 3600;
f_p_ch_hhv_total:       P_ch_HHV_total = P_ch_HHV_gas +
    P_ch_HHV_char + P_ch_HHV_tar;
```

Code 4.325: Enthalpy of formation

```
f_h_f_298_gas:  H_f_298_gas = (feed.Gas.gfhf273() + feed.Gas.gfht
    (25)) * massflow_gas / 3600;

ifl ref(feed.Dust) then
   f_h_f_298_dust:      H_f_298_dust = (feed.Dust.sfhf273() + feed
       .Dust.sfht(25)) * massflow_dust / 3600;
elsel
   f_h_f_298_dust2:     H_f_298_dust = 0;
endifl
ifl ref(feed.Char) then
   f_h_f_298_char:      H_f_298_char = feed.Char.hf298 *
       massflow_char / 3600;
elsel
   f_h_f_298_char2:     H_f_298_char = 0;
endifl
ifl ref(feed.Tar) then
   f_h_f_298_tar:       H_f_298_tar = feed.Tar.hf298 *
       massflow_tar / 3600;
elsel
   f_h_f_298_tar2:      H_f_298_tar = 0;
endifl

f_h_f_298_total:        H_f_298_total = H_f_298_gas + H_f_298_dust
    + H_f_298_char + H_f_298_tar;
```

The enthalpy of formation is calculated in Code 4.325 for the different stream components. If dust, char or tar are not present, the enthalpy of formation of the the missing components are set to zero. For gas the specific standard enthalpy of formation *gfhf273()* is a function-item in the gas global, dependent on the actual temperature and based on 0°C (273 K). As our desired standard enthalpy of formation *H_f_298_gas* is related to 25°C (298 K) the specific enthalpy (sensible heat) *feed.Gas.gfht(25)* -at 25°C, related to 0°C- corrects this inconsistency. The same correction also applies for dust. For char and tar the situation is simple. The variable-item *hf298* is defined in the organic global (Code 4.55) and calculates

the specific standard enthalpy for the current temperature related to 25°C (298 K). For the total enthalpy all the enthalpy-shares are summed up.

Code 4.326: Total enthalpy

```
f_h_total:                H_total = Q_298_total + H_f_298_total;
```

The definition of the total enthalpy (Code 4.326) is described in the script *Applied modelling in process engineering and energy technology* [104]. It is the sum of the enthalpy of formation and the sensible heat.

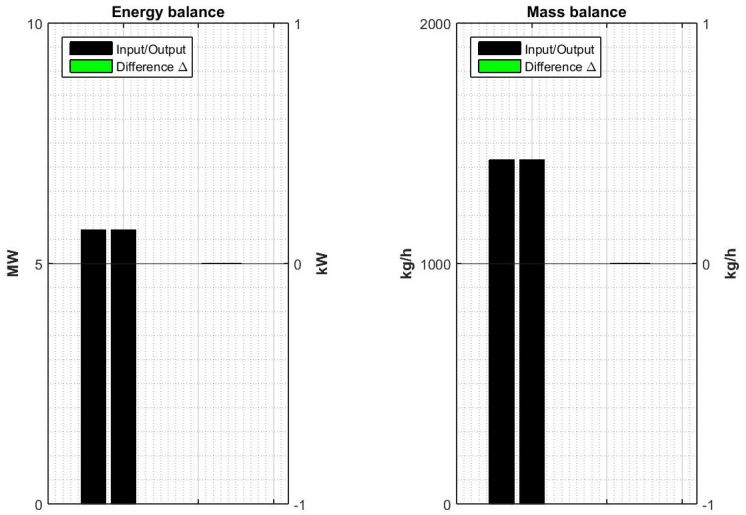


Figure 4.52: Energy and mass balance for the gas information-unit

Figure 4.52 proofs the unit's correct operation through an energy and mass balance.

Injector u_inj_so_

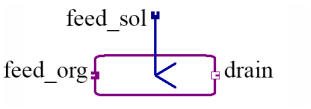


Figure 4.53: Injector library icon

The injection model injects one stream into an other. Both streams are of a different stream class, if not a mixer model has to be applied. For this explanation a solid stream *feed_sol* is injected into an organic one which enters at the connector *feed_org* and leaves as a mixture with solids at *drain*. This particular model has been picked as it is used for the simulations in Section 3.1. All the other ones are equal without the stream classes. For this and any other injection model, specification details are noted in the MDK-source file. In case of solid injection no adsorbed gas is allowed, while char is allowed in the solid stream.

Variables	Description
dp_org	Pressure drop for the organic the injector [bar]
E_loss	Exergy loss [kW]

Table 4.41: Injector model item description

Code 4.327: Energy balance

```
fh_total:        feed_org.h_total*feed_org.massflow + feed_sol.
    h_total*feed_sol.massflow = drain.h_total*drain.massflow;
```

Code 4.328: Water content

```
f_water:         feed_org.massflow*feed_org.water_content = drain.
    massflow*drain.water_content;
```

Code 4.329: Pressure drop organic stream

```
fp1:     feed_org.p = dp_org + drain.p;
```

Code 4.330: Linear combination of volume flows (additive mixing)

```
f_volflow:       drain.volflow = feed_org.volflow;
```

Code 4.331: Exergy loss

```
f_E_loss: feed_org.Exergy + feed_sol.Exergy = drain.Exergy +
    E_loss;
```

The sequence of codes (4.327-4.331) calculates variables in a way well known from other units. The only code that should be highlighted is Code 4.330. *volflow* is defined in the organic connection source code as total volume flow of an organic stream including organic substances, water and ash. The volume flow of these three components remain equal from the organic input to the output mixture due to Code 4.330.

Code 4.332: Composition of organic drain global

```
if1      ref(feed_org.Organic) != ref(drain.Organic)
          && ref(feed_sol.Char) && !ref(feed_sol.Gas)
then
f1wC:    drain.massflow*(1.0 - drain.water_content - drain.
    ash_content)*drain.Organic.wC = feed_org.massflow*(1.0 -
    feed_org.water_content - feed_org.ash_content)*feed_org.Organic
    .wC + feed_sol.massflow*(1.0 - feed_sol.char_content - feed_sol
    .gas_ads)*feed_sol.Char.wC;
f1wH:    drain.massflow*(1.0 - drain.water_content - drain.
    ash_content)*drain.Organic.wH = feed_org.massflow*(1.0 -
    feed_org.water_content - feed_org.ash_content)*feed_org.Organic
    .wH + feed_sol.massflow*(1.0 - feed_sol.char_content - feed_sol
    .gas_ads)*feed_sol.Char.wH;
f1wO:    drain.massflow*(1.0 - drain.water_content - drain.
    ash_content)*drain.Organic.wO = feed_org.massflow*(1.0 -
    feed_org.water_content - feed_org.ash_content)*feed_org.Organic
    .wO + feed_sol.massflow*(1.0 - feed_sol.char_content - feed_sol
    .gas_ads)*feed_sol.Char.wO;
f1wN:    drain.massflow*(1.0 - drain.water_content - drain.
    ash_content)*drain.Organic.wN = feed_org.massflow*(1.0 -
    feed_org.water_content - feed_org.ash_content)*feed_org.Organic
    .wN + feed_sol.massflow*(1.0 - feed_sol.char_content - feed_sol
    .gas_ads)*feed_sol.Char.wN;
f1wS:    drain.massflow*(1.0 - drain.water_content - drain.
    ash_content)*drain.Organic.wS = feed_org.massflow*(1.0 -
    feed_org.water_content - feed_org.ash_content)*feed_org.Organic
    .wS + feed_sol.massflow*(1.0 - feed_sol.char_content - feed_sol
    .gas_ads)*feed_sol.Char.wS;
f1wCl:   drain.massflow*(1.0 - drain.water_content - drain.
    ash_content)*drain.Organic.wCl = feed_org.massflow*(1.0 -
    feed_org.water_content - feed_org.ash_content)*feed_org.Organic
    .wCl + feed_sol.massflow*(1.0 - feed_sol.char_content -
    feed_sol.gas_ads)*feed_sol.Char.wCl;
endif1
if1      ref(feed_org.Organic) == ref(drain.Organic)
         && (!ref(feed_sol.Char) || ref(feed_sol.Char) == ref(
             feed_org.Organic))
         && !ref(feed_sol.Gas)
then
fmass:   feed_org.massflow * (1 - feed_org.water_content - feed_org
    .ash_content)  + feed_sol.massflow*feed_sol.char_content =
    drain.massflow * (1 - drain.water_content - drain.ash_content);
endif1
```

The first if-case of Code 4.332 assumes that entering and exiting organic objects are different and char is referenced in the solid stream. The partial mass balances for organic elements yields to organic composition and mass flow of the drain stream. For the second case no char is referenced, organic objects in *feed_org* and *drain* are the same.

Code 4.333: Mass balance inorganic components

```
if1 ref(drain.Ash) != ref(feed_org.Ash) || ref(drain.Ash) != ref(
    feed_sol.Solid) then
flashwAsh:      drain.massflow*drain.ash_content*drain.Ash.wAsh
          = feed_org.massflow*feed_org.ash_content*feed_org.Ash.
    wAsh      + feed_sol.massflow*feed_sol.Solid.wAsh;
flashwK2O:      drain.massflow*drain.ash_content*drain.Ash.wK2O
          = feed_org.massflow*feed_org.ash_content*feed_org.Ash.
    wK2O      + feed_sol.massflow*feed_sol.Solid.wK2O;
flashwMgO:      drain.massflow*drain.ash_content*drain.Ash.wMgO =
    feed_org.massflow*feed_org.ash_content*feed_org.Ash.wMgO      +
     feed_sol.massflow*feed_sol.Solid.wMgO;
flashwCaO:      drain.massflow*drain.ash_content*drain.Ash.wCaO
          = feed_org.massflow*feed_org.ash_content*feed_org.Ash.
    wCaO      + feed_sol.massflow*feed_sol.Solid.wCaO;
flashwSiO2:     drain.massflow*drain.ash_content*drain.Ash.wSiO2
          = feed_org.massflow*feed_org.ash_content*feed_org.Ash.
    wSiO2     + feed_sol.massflow*feed_sol.Solid.wSiO2;
flashwOlivine:  drain.massflow*drain.ash_content*drain.Ash.
    wOlivine      = feed_org.massflow*feed_org.ash_content*feed_org.
    Ash.wOlivine  + feed_sol.massflow*feed_sol.Solid.wOlivine;
flashwCaCO3:    drain.massflow*drain.ash_content*drain.Ash.wCaCO3
          = feed_org.massflow*feed_org.ash_content*feed_org.Ash.
    wCaCO3    + feed_sol.massflow*feed_sol.Solid.wCaCO3;
flashwDolomite: drain.massflow*drain.ash_content*drain.Ash.
    wDolomite     = feed_org.massflow*feed_org.ash_content*feed_org.
    Ash.wDolomite          + feed_sol.massflow*feed_sol.Solid.
    wDolomite;
flashwCaSO4:    drain.massflow*drain.ash_content*drain.Ash.wCaSO4
          = feed_org.massflow*feed_org.ash_content*feed_org.Ash.
    wCaSO4    + feed_sol.massflow*feed_sol.Solid.wCaSO4;
flashwCaOH2:    drain.massflow*drain.ash_content*drain.Ash.wCaOH2
          = feed_org.massflow*feed_org.ash_content*feed_org.Ash.
    wCaOH2    + feed_sol.massflow*feed_sol.Solid.wCaOH2;

elsel
f2ash:   drain.massflow*drain.ash_content          = feed_org.
    massflow*feed_org.ash_content          + feed_sol.massflow;
endifl
```

For the first if-condition,the ash global in the drain stream is different to the one in the feed stream. The second option considers all inorganic globals to be equal, see Code 4.333.

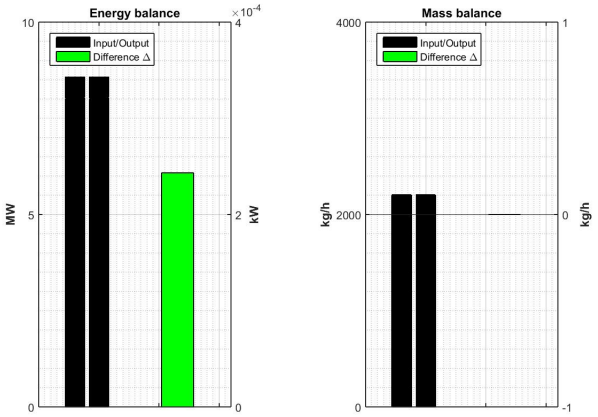


Figure 4.54: Energy and mass balance for the injector unit

Figure 4.54 proofs the unit's correct operation through an energy and mass balance.

Loop connector u_loop_w_

Figure 4.55: Loop connector library icon

The loop connector model is available for all the relevant four stream classes (not for shaft-connections). Like for the injector model, this model has been picked as it is used for the simulation of Section 3.1. The model is applied to close a circulating stream. Water is fed at *feed* and leaves at *drain*. Drain pressure and temperature are equal to the feed values, see Code 4.334 and 4.335. No separate mass balance is considered and no items are installed for this model.

Code 4.334: Continuity of enthalpy

```
f_h:      drain.h = feed.h;
```

Code 4.335: Continuity of pressure

```
f_press:          drain.p = feed.p;
```

Trivial, Figure 4.56 shows the unit's energy and mass balance.

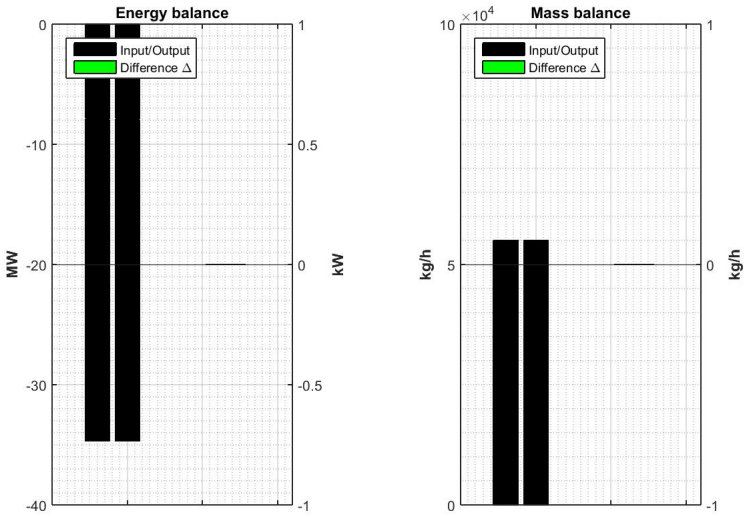


Figure 4.56: Energy and mass balance for the water loop unit

Mixer gas u_mixer_g_

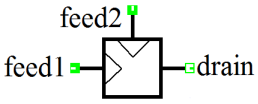


Figure 4.57: Mixer library icon

Also the mixer model is available for the four different stream classes. All of them are applied in the simulation of Section 3.1. The most elaborate source code has the gaseous model and therefore it has been chosen to be described. Gas is fed via entrance *feed1* and *feed2*. The mixture of both inlet stream leaves through the exit *drain*. Additional theoretical and explanatory information can be found in [76][5.1.5].

Variables	Description
dp_1	Pressure drop of the stream attached to *feed1* [bar]
dp_2	Pressure drop of the stream attached to *feed2* [bar]
E_loss	Exergy loss [kW]

Table 4.42: Gas mixer model item description

Code 4.336: Mass balance

```
fmass:   feed1.massflow + feed2.massflow = drain.massflow;
```

Code 4.337: Pressure drop feed1

```
fp1:     feed1.p = dp_1 + drain.p;
```

Code 4.338: Pressure drop feed1

```
fp2:    feed2.p = dp_2 + drain.p;
```

Code 4.339: Exergy loss

```
f_E_loss:       feed1.Exergy + feed2.Exergy = drain.Exergy +
    E_loss;
```

Nothing new considering the mass balance, pressure drops or exergy loss calculations in Code 4.336 to 4.339.

Code 4.340: Drain gas global composition

```
if1     ref(feed1.Gas) != ref(feed2.Gas)
        && ref(feed1.Gas) != ref(drain.Gas)
        && ref(feed2.Gas) != ref(drain.Gas)
then
        f1wAr:  drain.massflow * drain.Gas.wAr  = feed1.massflow *
            feed1.Gas.wAr       + feed2.massflow * feed2.Gas.wAr;
        f1wC2H4:        drain.massflow * drain.Gas.wC2H4        =
            feed1.massflow * feed1.Gas.wC2H4      + feed2.massflow
            * feed2.Gas.wC2H4;
                        .
                        .
        f1wO2:  drain.massflow * drain.Gas.wO2  = feed1.massflow *
            feed1.Gas.wO2       + feed2.massflow * feed2.Gas.wO2;
        f1wSO2: drain.massflow * drain.Gas.wSO2 = feed1.massflow *
            feed1.Gas.wSO2      + feed2.massflow * feed2.Gas.wSO2
            ;
endif1
```

To generate the *drain* stream composition, first of all the composition of the gas global is investigated. Four different cases are considered. If all three streams have the same gas global, no additional equations are required.
If all three gas globals (*feed1*, *feed2*, *drain*) are different, Code 4.340 is applied. It shows the species balance for all gaseous components except water (first case).
The second case describes the scenario that both feed streams use the same gas global and the *drain* gas is different.
For the last case *feed1* and *drain* use the same gas global while *feed2* uses a different gas global or *feed2* and *drain* use the same gas global while *feed1* uses a different gas global.
All these scenarios of the same or different globals are considered for dust, char and tar and can be seen in the units MDK-file.

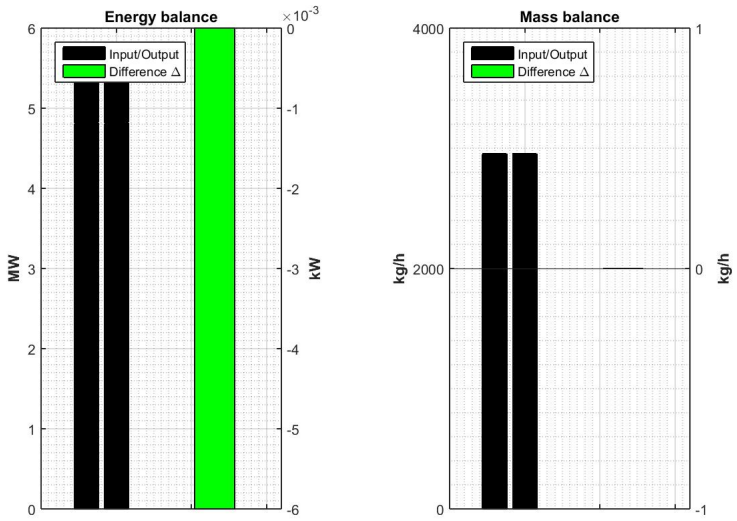


Figure 4.58: Energy and mass balance for the mixer unit

Figure 4.58 presents the energy and mass balance of this unit.

Electric motor u_mot_el_

Figure 4.59: Electric motor library icon

Due to the selected rotation mode of the model's terminals, the connector from the left side is always the input connector and on the right side there is always the output, even when rotating the icon. The input is called *shaft_in* and its output *shaft_out*.

Variables	Description
power	Electrical power [kW]
E_loss	Exergy loss [kW]
Parameters	**Description**
eta_el	Electrical efficiency [%]
eta_m	Mechanical efficiency [%]

Table 4.43: Electric motor model item description

Code 4.341: Shaft only from left

```
if1 ref(shaft_in) && !ref(shaft_out) then
        f1a:    (eta_el*0.01)*(eta_m*0.01)*power + shaft_in.power
            = 0.0;
        f_E_loss_a:     shaft_in.power=E_loss-power;
endif1
```

Code 4.342: Shaft only from right

```
if1 !ref(shaft_in) && ref(shaft_out) then
        f1b:    (eta_el*0.01)*(eta_m*0.01)*power - shaft_out.power
            = 0.0;
        f_E_loss_b:     shaft_out.power+E_loss-power=0;
endif1
```

Code 4.343: Shaft from both sides

```
if1 !ref(shaft_in) && ref(shaft_out) then
        f1b:    (eta_el*0.01)*(eta_m*0.01)*power - shaft_out.power
            = 0.0;
        f_E_loss_b:     shaft_out.power+E_loss-power=0;
endif1
```

Code 4.341 considers all possible cases of connecting the electric motor. As an mass and energy balance do not have informative significance it is not presented.

Organic Rankine cycle ORC u_orc_2st_

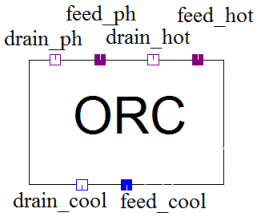


Figure 4.60: ORC library icon

This model performs the conversion of heat to electric energy where the heat carrier is an organic substance, the cooling is done by water. The actual electric efficiency is compared to the theoretically possible efficiency of the Carnot-cycle. Due to the selected rotation mode of the model's terminals, the connector from the left side is always the input connector and on the right side there is always the output, even when rotating the icon. The input is called *shaft_in* and its output *shaft_out*. Further information on the ORC process model is provided by T. Pröll [76][5.6].

Switch	Description
Type	Switch allows the modelling of special performance curves for different types of ORC units. In the case ”Unspecific”, no performance correlation applies the other ones are TurbodenOW and User_defined
Variables	**Description**
dp_ph	Pressure drop of preheating liquid [bar]
dp_hot	Pressure drop of heating liquid [bar]
dp_cold	Pressure drop of cooling liquid [bar]
T_evap	Temperature level of heat supply to rankine cycle (practically evaporation temperature) [°C]
T_cond	Temperature level of heat withdrawal from rankine cycle (practically condensation temperature) [°C]
bias_T_hot_in_out	Bias between evaporation temperature of the Rankine cycle and the hot side heat supply in/out. Value between 0 and 1: 0: T_evap = heat carrier temperature outlet (cold) 1: T_evap = heat carrier temperature inlet (hot)
bias_T_cold_in_out	Bias between condensation temperature of the Rankine cycle and the cold side heat decoupling in/out. Value between 0 and 1: 0: T_cond = cooling water inlet temperature (cold) 1: T_cond = cooling water outlet temperature (hot)
Q_ph	Heat from preheat cycle [kW]
Q_evap	Heat to evaporator = heat from hot organic cycle (= Q_hot) [kW]
Q_total_in	Total heat to ORC (Q_evap + Q_ph) [kW]
prop_Q_total_in	Proportion of Q_evap to Q_ph
Q_total_nom	Nominal heat total introduced on hot and preheating side [kW]
load_factor	Specifies the heat load on the ORC [%]
P_el	Electric power of the ORC/generator unit [kW]
eta_el	Actual electric efficiency reached by the ORC module [%]
eta_carnot	Hypothetical Carnot-cycle efficiency between highest and lowest temperature of hot and cold side [%]
eff_rel_carnot	Efficiency of the ORC relative to hypothetical Carnot-cycle [%]
Q_out	Exported heat in cooling water [kW]
eta_Q	Heat decoupling efficiency [%]
Q_loss	Heat loss of the ORC setup to the environment including heat produced by friction and electric irreversibly [kW]
q_loss_rel	Heat loss related to introduced heat Q_in [%]
E_loss	Exergy loss [kW]

Table 4.44: ORC model item description

Code 4.344: Energy balance

```
fEnergy:        Q_out + P_el + Q_loss = Q_evap + Q_ph;
```

Code 4.345: Heat introduced on hot side

```
fQ_evap:          Q_evap*3600.0 = feed_hot.massflow*feed_hot.h_total
    - drain_hot.massflow*drain_hot.h_total;
```

Code 4.346: Heat introduced for media preheating

```
fQ_ph:  Q_ph*3600.0 = feed_ph.massflow*feed_ph.h_total - drain_ph.
    massflow*drain_ph.h_total;
```

Code 4.347: Total heat introduced

```
fQ_total_in:    Q_total_in = Q_evap + Q_ph;
```

Code 4.348: Proportion of Q_evap to Q_ph

```
fprop_Q_total_in: prop_Q_total_in = Q_evap/Q_ph;
```

Code 4.349: Heat exported to cooling water

```
fQ_out: Q_out*3600.0 = drain_cold.massflow*drain_cold.h_total -
    feed_cold.massflow*feed_cold.h_total;
```

Code 4.350: Heat loss related to Q_in

```
fQ_loss:          Q_loss = (q_loss_rel/100.0)*(Q_evap + Q_ph);
```

The various heat values of Table 4.44 are defined by Code 4.344 to 4.350.

Code 4.351: Electric efficiency

```
feta_el:          P_el = (eta_el/100.0)*(Q_evap + Q_ph);
```

Code 4.352: Heat decoupling efficiency

```
feta_Q: 0.01*eta_Q = Q_out/(Q_evap + Q_ph);
```

Code 4.353: Hypothetical Carnot-cycle efficiency based on actual evaporation and condensation temperature of the Rankine cycle

```
feta_carnot:      eta_carnot/100.0 = 1.0 - (T_cond + 273.15)/(T_evap
    + 273.15);
```

Code 4.354: Efficiency of ORC setup relative to hypothetical Carnot-cycle efficiency

```
feff_rel_carnot:          eta_el = (eff_rel_carnot/100.0)*eta_carnot
    ;
```

Efficiency coefficients are calculated from Code 4.351 to 4.354. The Carnot efficiency of Code 4.353 is defined as the obtained usable work divided through the supplied heat. Out of this definition its final form of Equation 4.11 is created.

$$\eta_C = 1 - \frac{T\_cond}{T\_evap} \tag{4.11}$$

The temperatures are set in Kelvin while T_cond to T_evap represents the lowest to the highest temperature of the process.

Code 4.355: Temperature bias

```
fbias_T_hot_in_out:      T_evap = drain_hot.t + (feed_hot.t -
    drain_hot.t)*bias_T_hot_in_out;
fbias_T_cold_in_out:     T_cond = feed_cold.t + (drain_cold.t -
    feed_cold.t)*bias_T_cold_in_out;
```

The variables $bias\_T\_hot\_in\_out$ and $bias\_T\_cold\_in\_out$ -of Code 4.355- describe the location of the evaporation and the condensation temperature levels between the heat carrier inlet/outlet temperatures.

Code 4.356: Mass balance equations

```
fmass_hot:       feed_hot.massflow = drain_hot.massflow;
fmass_ph:    feed_ph.massflow = drain_ph.massflow;
fmass_cold:      feed_cold.massflow  = drain_cold.massflow;
```

Code 4.357: Mass balance for water in organic streams

```
fwater_cont_hot: feed_hot.water_content = drain_hot.water_content;
fwater_cont_ph: feed_ph.water_content = drain_ph.water_content;
```

Code 4.358: Ash mass balance for hot organic stream

```
ifl ref(feed_hot.Ash) || ref(drain_hot.Ash) then
        fash_cont_hot:  feed_hot.ash_content = drain_hot.
            ash_content;
endifl
```

Mass balances for all the streams are defined by Code 4.356 and in particular for the water content of the organic streams and the ash content for the hot stream are specified by Code 4.357 and 4.358.

Code 4.359: Density of streams

```
f_dens1:         drain_hot.rho = feed_hot.rho;
f_dens2:         drain_ph.rho = feed_ph.rho;
```

Code 4.360: Performance correlations

```
ifl Type == TurbodenOW  then f1perf:     eta_el = -1.4154e-6*(
    Q_evap + Q_ph)*(Q_evap + Q_ph) + 8.4421e-3*(Q_evap + Q_ph) +
    2.6976;        endifl
ifl Type == Unspecific  then f2perf:     eta_el = -0.001998*
    load_factor*load_factor + 0.3881*load_factor - 0.95;
    endifl
```

Code 4.361: Pressure drops

```
fdp_hot:         feed_hot.p - dp_hot = drain_hot.p;
fdp_ph:          feed_ph.p - dp_ph = drain_ph.p;
fdp_cold:        feed_cold.p - dp_cold = drain_cold.p;
```

Code 4.362: Exergy loss

```
f_E_loss: E_loss = feed_hot.Exergy + feed_cold.Exergy + feed_ph.
    Exergy - drain_hot.Exergy - drain_cold.Exergy - drain_ph.Exergy
    - P_el;
```

The density of the hot and ph-stream are defined by Code 4.359. The electrical efficiency is calculated in different ways depending on the chosen switch option (Code 4.360). The pressure drops are calculated by Code 4.361 and the exergy loss by Code 4.362. Figure

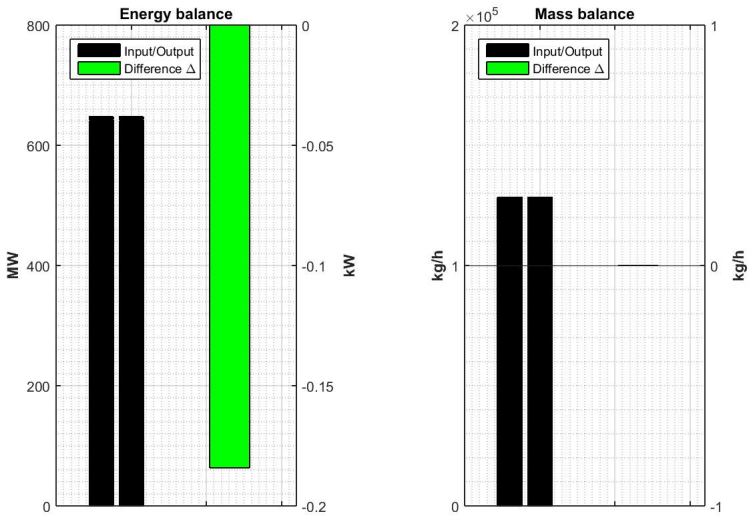


Figure 4.61: Energy and mass balance for the ORC model

4.61 shows the energy and mass balance of this unit.

Gas oxidation reactor u_oxid_react_

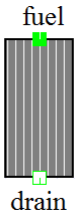


Figure 4.62: Gas oxidation reactor library icon

This is a model for thermal or catalytic gas oxidation that allows permitting dust, char, and tar in gaseous fuel. No conversions of inorganic solids takes place. The assumptions of no char or tar in the flue gas drain stream are made. If dust is referenced in the fuel, it also must be referenced in the drain stream. The gas is fed through the *fuel* terminal and leaves through the *drain* exit.

Variables	Description
dp	Pressure drop from fuel gas to drain [bar]
lambda	Air ratio of the combustion chamber based on the amount of oxygen in air stream needed for stoichiometric combustion of all substances entering the combustion chamber (e.g. also combustables entering with the air stream) [-]
CO_slip	Molar ratio between CO and CO_2 in flue gas. This value must be determined by the system (set or exiting CO concentration set) if lambda ≥ 1.0. If lambda < 1.0 the CO_slip has no influence on the system and must be set to a dummy value [% CO]
X_CO	Conversion of CO in the oxidation reactor [%]
P_th	Thermal fuel power entering the oxidation reactor [kW]
q_loss_rel	Relative heat loss of the combustion reactor in % of thermal power [%]
Q_loss	Heat loss of the combustion reactor [kW]
E_loss	Exergy loss [kW]

Table 4.45: Gas oxidation model item description

Code 4.363: Energy balance

```
fEnergy:          drain.massflow*drain.h_total + Q_loss*3600.0 =
    fuel.massflow*fuel.h_total;
```

Code 4.364: Thermal power of reactor

```
ifl !ref(fuel.Char) && !ref(fuel.Tar) then
        fP_th_1: P_th*3.6 = fuel.nvolflow*fuel.Gas.LHV;
endifl
ifl ref(fuel.Char) && !ref(fuel.Tar) then
        fP_th_2: P_th*3600.0 = fuel.nvolflow*(fuel.Gas.LHV*1000.0
           + (fuel.char_content/1000.0)*fuel.Char.lhv);
endifl
ifl !ref(fuel.Char) && ref(fuel.Tar) then
        fP_th_3: P_th*3600.0 = fuel.nvolflow*(fuel.Gas.LHV*1000.0
           + (fuel.tar_content/1000.0)*fuel.Tar.lhv);
endifl
ifl ref(fuel.Char) && ref(fuel.Tar) then
        fP_th_4: P_th*3600.0 = fuel.nvolflow*(fuel.Gas.LHV*1000.0
           + (fuel.char_content/1000.0)*fuel.Char.lhv + (fuel.
           tar_content/1000.0)*fuel.Tar.lhv);
endifl
```

For the calculation of the thermal power, four different scenarios are considered. First, neither char nor tar are assumed in the *fuel* stream. For the second case, char but no tar is in the entrance stream. The third case considers tar but no char in the *fuel* stream. For the last case both, char and tar, are in the inlet gas.

Code 4.365: Heat loss of the combustion reactor

```
fQ_loss: Q_loss = (fuel.massflow*fuel.h + (fuel.nvolflow*((fuel.
    dust_content/1000.0)*fuel.h_dust + (fuel.char_content/1000.0)*
```

```
    fuel.h_char + (fuel.tar_content/1000.0)*fuel.h_tar)))*(
    q_loss_rel/100.0)/3600.0;
```

Code 4.366: Exergy loss due to irreversible combustion and heat loss to the environment

```
fE_loss:        drain.Exergy + E_loss = fuel.Exergy;
```

Code 4.367: Pressure drop

```
fdp:    fuel.p = drain.p + dp;
```

Heat loss, exergy loss and pressure drop are calculated via Code 4.365 to 4.367

Code 4.368: Elementary mass balances

```
fAr_1:  drain.massflow*drain.Gas.wAr = fuel.massflow*fuel.Gas.wAr;
fC_1:   drain.massflow*12.011*(2*drain.Gas.wC2H4/28.0536 + 2*drain
    .Gas.wC2H6/30.0694 + 3*drain.Gas.wC3H8/44.0962 + drain.Gas.wCH4
    /16.0428 + drain.Gas.wCO/28.0104 + drain.Gas.wCO2/44.0098 +
    drain.Gas.wHCN/27.02568) = fuel.massflow*12.011*(2*fuel.Gas.
    wC2H4/28.0536 + 2*fuel.Gas.wC2H6/30.0694 + 3*fuel.Gas.wC3H8
    /44.0962 + fuel.Gas.wCH4/16.0428 + fuel.Gas.wCO/28.0104 + fuel.
    Gas.wCO2/44.0098 + fuel.Gas.wHCN/27.02568);
                            .
                            .
fCl_1:  drain.massflow*35.4527*drain.Gas.wHCl/36.46064 = fuel.
    massflow*35.4527*fuel.Gas.wHCl/36.46064;
endifl
```

To evaluate the *drain* stream composition of the gas global, an elementary mass balance for all the considered elements is applied. Four different scenarios are thought through, Code 4.368 neither considers char nor tar in the *fuel* stream. Other scenarios are, char is present but tar is not or vice versa and another case assumes both to be present in the *fuel* stream.

Code 4.369: Inorganic components

```
ifl ref(fuel.Dust) && ref(drain.Dust)
        && ref(fuel.Dust) == ref(drain.Dust)
then
        fdust_cont:     fuel.nvolflow*fuel.dust_content = drain.
            nvolflow*drain.dust_content;
endifl

ifl ref(fuel.Dust) && ref(drain.Dust)
        && ref(fuel.Dust) != ref(drain.Dust)
then
fAsh:   fuel.nvolflow*fuel.dust_content*fuel.Dust.wAsh = drain.
    nvolflow*drain.dust_content*drain.Dust.wAsh;
fK2O:   fuel.nvolflow*fuel.dust_content*fuel.Dust.wK2O = drain.
    nvolflow*drain.dust_content*drain.Dust.wK2O;
                            .
                            .
```

```
fCaOH2: fuel.nvolflow*fuel.dust_content*fuel.Dust.wCaOH2 = drain.
    nvolflow*drain.dust_content*drain.Dust.wCaOH2;
endifl
```

The composition of the *drain* stream -considering its inorganic components- is obtained by Code 4.369. For the first case, it is assumed that *fuel* and *drain* stream are carrying the same dust global. For the second case, different dust globals are assumed for the in- and outlet. In this case the drain.dust_content is determined by the complete set of elementary balances of Code 4.369 together with Code 4.57, which applies the sum of all inorganic mass fractions to be one.

Code 4.370: Air ratio lambda

```
flambda: lambda*drain.massflow*(drain.Gas.wO2 - drain.Gas.wCO
    *31.9988/56.0208) = fuel.massflow*fuel.Gas.wO2*(lambda - 1.0);
```

Code 4.371: No gaseous organic carbon in flue gas

```
frC2H4: drain.Gas.yC2H4 = 0.0;
frC2H6: drain.Gas.yC2H6 = 0.0;
frC3H8: drain.Gas.yC3H8 = 0.0;
frCH4:  drain.Gas.yCH4 = 0.0;
```

The ratio of available oxygen to stoichiometric required oxygen is calculated by Code 4.370. Also relevant for the drain gas composition is Code 4.371. It is assumed that the listed gaseous components are not present in the *drain* stream.

Code 4.372: CO slip

```
frCO:   if lambda >= 1.0        then
drain.Gas.yCO = (CO_slip/100.0)*drain.Gas.yCO2;
else  drain.Gas.yO2 = 0.5*(CO_slip/100.0)*drain.Gas.yCO2;
```

Code 4.373: CO conversion

```
frX_CO: X_CO = 100.0*(fuel.massflow*fuel.Gas.wCO - drain.massflow*
    drain.Gas.wCO)/(fuel.massflow*fuel.Gas.wCO);
```

Carbon monoxide slip and its conversion are considered for the *drain* gas composition through Code 4.372 and 4.373.

Code 4.374: Flue gas composition assumptions

```
frH2:   drain.Gas.yH2 = 0.0;
frH2S:  drain.Gas.yH2S = 0.0;
frHCN:  drain.Gas.yHCN = 0.0;
frN2O:  drain.Gas.yN2O = 0.0;
frNH3:  drain.Gas.yNH3 = 0.0;
```

Code 4.375: No thermal NO is formed

```
frN2: drain.nvolflow*drain.Gas.yN2 = fuel.nvolflow*fuel.Gas.yN2;
```

To complete the *drain* gas composition the assumtions of Code 4.374 are 4.375 are made. None component of Code 4.374 is present in the *drain* gas stream and due to Code 4.375, N_2 passes through the reactor without reacting.

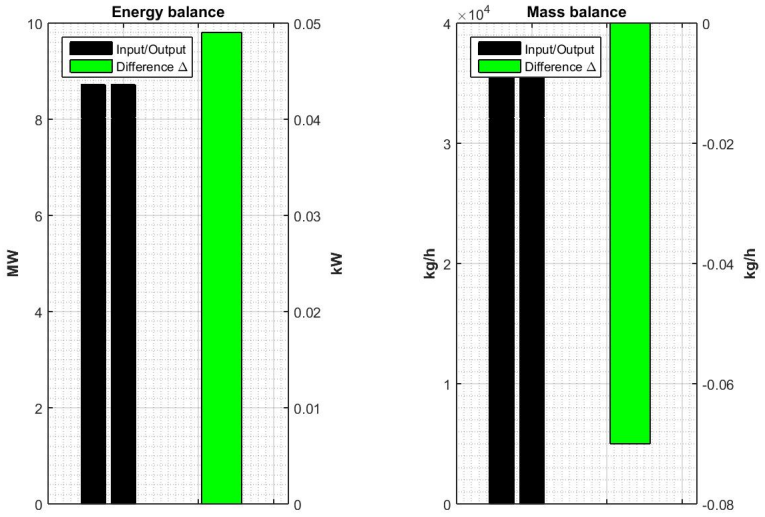


Figure 4.63: Energy and mass balance for the gas oxidation reactor

Figure 4.63 is a presentation of the energy and mass balance of this unit.

Pipe u_pipe_gas_

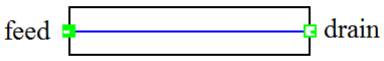


Figure 4.64: Pipe library icon

This model simulates a gas pipe with pressure drop and possible heat loss to the environment. A pipe model is also available for organic and water streams. For all of them the inlet is called *feed* and the outlet *drain*.

Variables	Description
dp	Pressure drop in pipe [bar]
dt	Temperature drop in pipe [°C]
q_loss_rel	Loss of heat relative to sensible gas heat in feed stream [%]
Q_loss	Heat loss [kW]
E_loss	Exergy loss [kW]
dp_0	Initial pressure drop [bar]
opvolflow_0	Initial volume flow [m^3/h]
v_0	Initial specific volume [m^3/kg]

Table 4.46: Pipe model item description

Code 4.376: Energy balance

```
fh_total:        drain.massflow*drain.h_total + Q_loss*3600.0 =
    feed.massflow*feed.h_total;
```

Code 4.377: Mass balances

```
f_mass: drain.massflow = feed.massflow;
ifl ref(feed.Dust) && ref(drain.Dust) then
f_dust_content: drain.dust_content = feed.dust_content; endifl
ifl ref(feed.Char) && ref(drain.Char) then
f_char_content: drain.char_content = feed.char_content; endifl
ifl ref(feed.Tar) && ref(drain.Tar) then
f_tar_content:  drain.tar_content = feed.tar_content;   endifl
```

Code 4.378: Pressure drop

```
fdp:     feed.p = drain.p + dp;
```

Code 4.379: Temperature drop

```
fdt:     feed.t = drain.t + dt;
```

Code 4.380: Heat loss

```
fq_loss:         Q_loss*3600.0 = feed.massflow*feed.h*q_loss_rel
    /100.0;
```

Code 4.381: Exergy loss

```
fExergy:        drain.Exergy + E_loss = feed.Exergy;
```

The mass balances of different components are fulfilled for the unit due to 4.377. From Code 4.378 to 4.381, variables of Table 4.46 are calculated.

Code 4.382: Variables for part load model

```
fident_dp_0:      dp_0 = dp;
fident_opvolflow_0: opvolflow_0 = (feed.opvolflow + drain.
    opvolflow)/2;
fident_v0:        v_0 = (feed.v + drain.v)/2;
```

The variables from Code 4.382 are just defined for the sub-model that simulates a pipe at part load behavior. They are not relevant for this pipe model.

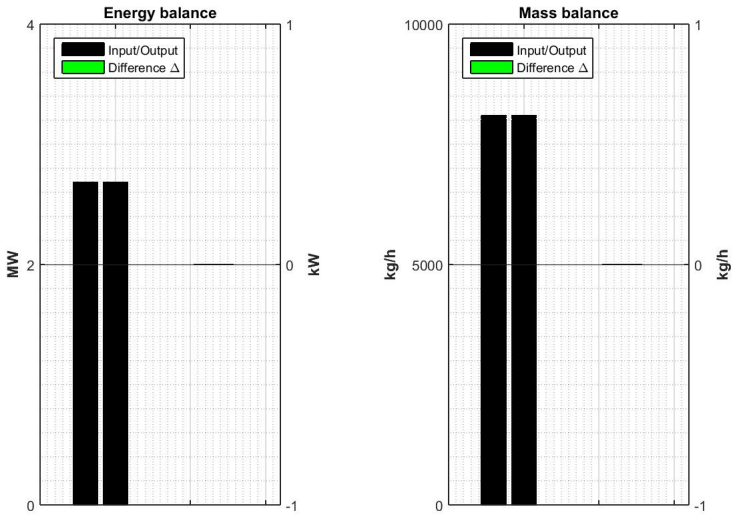


Figure 4.65: Energy and mass balance for the pipe model

Figure 4.65 presents the energy and mass balance of the pipe model.

Pump u_pump_w_

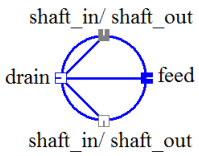


Figure 4.66: Pump library icon

Besides for water, the pump model is also available for the organic stream class. Due to the rotation mode for the shaft-terminals, the shaft-inlet is always connected from above even when rotating the icon. Like for many other units, connected with a shaft, this is the reason why both shaft connector have the equal name *shaft_in* and *shaft_out*. The water stream enters at the connector *feed* and exits at *drain*. Additional information can be found in [76][5.1.2].

Variables	Description
dp	Pressure difference between drain and feed [bar]
press_ratio	Pressure ratio p_drain/p_feed [-]
eta_s	Isentropic efficiency of the compression process (dissipation in the fluid) [%]
eta_m	Mechanical efficiency of the pump (bearing friction, etc.) [%]
P_rotor_disc	Power at the rotor disc, which will be absorbed by the fluid [kW]
E_loss	Exergy loss [kW]

Table 4.47: Pump model item description

Code 4.383: Pressure increase

```
fdp:    drain.p = feed.p + dp;
```

Code 4.384: Mass balance

```
f1:     feed.massflow = drain.massflow;
```

Code 4.385: Polytropic compression

```
f2:     feed.s = drain.wfsph(drain.p, feed.h + (drain.h - feed.h)*
    eta_s/100.0);
```

Code 4.386: Energy balance and exergy loss

```
if1 ref(shaft_in) && ref(shaft_out) then
f1h:    (drain.h - feed.h)*feed.massflow/(eta_m/100) = (shaft_in.
    power - shaft_out.power)*3600.0;
f1_E_loss: feed.Exergy + shaft_in.power = E_loss + drain.Exergy +
    shaft_out.power;
```

```
endifl

ifl ref(shaft_in) && !ref(shaft_out) then
f2h: (drain.h - feed.h)*feed.massflow/(eta_m/100) = shaft_in.power
    *3600.0;
f2_E_loss: feed.Exergy + shaft_in.power = E_loss + drain.Exergy;
endifl

ifl !ref(shaft_in) && ref(shaft_out) then
f3h: (drain.h - feed.h)*feed.massflow/(eta_m/100)  = - shaft_out.
    power*3600.0;
f3_E_loss: feed.Exergy = E_loss + drain.Exergy + shaft_out.power;
endifl
```

Pressure drop, mass balance and the polytropic efficiency are defined by Code 4.383 to 4.385. The energy and mass balance are considered for the cases that both shaft terminals are connected, top shaft respectively the bottom shaft terminal is just connected, see Code 4.386.

Code 4.387: Power of rotor disc

```
f1:       feed.massflow = drain.massflow;
```

The variable P_rotor_disc is defined according to Table 4.47. In practice the variable represents sensitive heat input to the fluid.

Code 4.388: Pressure ratio

```
fpress_ratio:    feed.p*press_ratio = drain.p;
```

Code 4.388 defines the ratio between the feed pressure and the drain pressure.

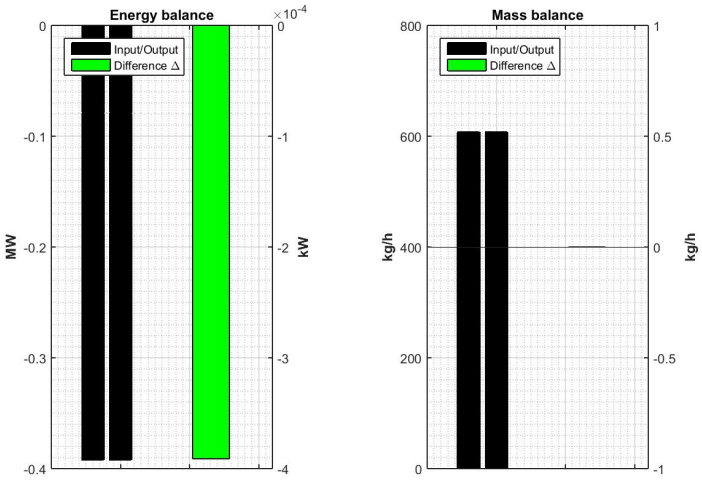


Figure 4.67: Energy and mass balance for the pump unit

Figure 4.67 is an illustration of a calculated energy and mass balance for this model.

Externally heated pyrolyzer u_pyrol_htd_

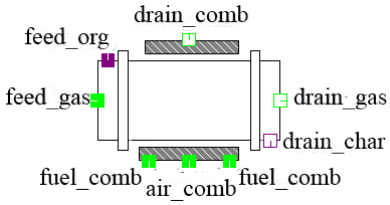


Figure 4.68: Externally heated reactor library icon

The model contains two gaseous outlet terminals, which is quite uncommon. Therefore, the model solves the mass and energy balances separately for the pyrolyzer (*drain_gas*) and the combuster (*drain_comb*) with external heating. The only link between the external heating section (gas combustor with dust) and the pyrolyzer is the transferred heat *Q_trans*. The *feed_gas* and *air_comb* are pure gas streams, no dust, char, or tar is allowed. It is assumed that *drain_gas* and *feed_comb* contain all of the dust, char, and tar. The *drain_comb* stream just contains dust but no char or tar. While *feed_org* and *drain_char* contain ash.

Variables	Description
dp_air_comb	Pressure drop from fluidization to producer gas drain [bar]
dp_air_rel_comb	Relative pressure drop for air based on feed pressure [bar]
dp_fuel_comb	Pressure drop from fuel gas to drain [bar]
dp_fuel_rel_comb	Relative pressure drop for fuel based on feed pressure [bar]
P_th_comb	Thermal power of the combustion chamber based on lower heating values of all combustibles entering the reactor [kW]
lambda_comb	Air ratio of the combustion chamber based on the amount of oxygen in air stream needed for stoichiometric combustion of all substances entering the combustion chamber (e.g. also combustables entering with the air stream) [-]
CO_slip_comb	Molar ratio between CO and CO_2 in flue gas. This value must be determined by the system (set or exiting CO concentration set) if lambda ≥ 1.0. If lambda < 1.0 the CO_slip has no influence on the system and must be set to a dummy value [%CO_2]
Q_loss_comb	Heat loss of the external heating shell [kW]
q_loss_rel_comb	Relative heat loss of the external heating shell in % of thermal fuel power of the combustion chambers [%]
Q_trans	Heat transfer power from external heating to pyrolyser [kW]
dp_feed_org	Pressure drop in pyrolysis reactor: organic feed to gas drain [bar]
dp_feed_org_rel	Relative pressure drop in pyrolysis reactor: organic feed to gas drain [%]
dp_feed_gas	Pressure drop in pyrolysis reactor: gas feed to gas drain [bar]
dp_feed_gas_rel	Relative pressure drop in pyrolysis reactor: gas feed to gas drain [%]
dp_drain_gas_char	Pressure difference of drain streams: dp = p_gas - p_char [bar]
dt_drain_gas_char	Temperature difference between gas and char drain [°C]

Variables	Description
P_th_pyr	Thermal power of the fuel entering the pyrolyzer based on lower heating value [kW]
fract_char_bed	Mass fraction of ash free and water free fuel exiting the pyrolyzer as ungasified char with the bed material [kg/kg]
fract_tar_pg	Mass fraction of the ash free and water free fuel exiting the pyrolyzer as tar in the producer gas [kg/kg]
fract_fly_ash	Mass fraction of ash introduced with the fuel, which is exiting the pyrolyzer as dust in the producer gas [kg/kg]
unconv_C	Conversion of carbon to gaseous compounds (gas and tar), solid carbon in both fly char and char drain in considered unconverted carbon [%]
E_loss	Exergy loss [kW]

Table 4.48: Externally heated pyrolyzer model item description

Combustion reactor

Code 4.389: Energy balance of the combustion chamber

```
fh_total_comb:  drain_comb.massflow*drain_comb.h_total + (
    Q_loss_comb + Q_trans)*3600.0  = air_comb.massflow*air_comb.
    h_total + fuel_comb.massflow*fuel_comb.h_total;
```

Code 4.390: Thermal power of the combustion reactor

```
ifl !ref(fuel_comb.Char) && !ref(fuel_comb.Tar) then
fP_th_comb_1:   P_th_comb*3600.0        = air_comb.nvolflow*
    air_comb.Gas.LHV*1000.0 + fuel_comb.nvolflow*fuel_comb.Gas.LHV
    *1000.0;
endifl
ifl ref(fuel_comb.Char) && !ref(fuel_comb.Tar) then
fP_th_comb_2:   P_th_comb*3600.0        = air_comb.nvolflow*
    air_comb.Gas.LHV*1000.0 + fuel_comb.nvolflow*(fuel_comb.Gas.LHV
    *1000.0 + (fuel_comb.char_content/1000.0)*fuel_comb.Char.lhv);
endifl
ifl !ref(fuel_comb.Char) && ref(fuel_comb.Tar) then
fP_th_comb_3:   P_th_comb*3600.0        = air_comb.nvolflow*
    air_comb.Gas.LHV*1000.0 + fuel_comb.nvolflow*(fuel_comb.Gas.LHV
    *1000.0 + (fuel_comb.tar_content/1000.0)*fuel_comb.Tar.lhv);
endifl
ifl ref(fuel_comb.Char) && ref(fuel_comb.Tar) then
fP_th_comb_4:   P_th_comb*3600.0        = air_comb.nvolflow*
    air_comb.Gas.LHV*1000.0 + fuel_comb.nvolflow*(fuel_comb.Gas.LHV
    *1000.0 + (fuel_comb.char_content/1000.0)*fuel_comb.Char.lhv +
    (fuel_comb.tar_content/1000.0)*fuel_comb.Tar.lhv);
endifl
```

The energy balance of the combustion chamber is documented by Code 4.389. Its thermal power is calculated via Code 4.390. It is distinguished between four different scenarios. At first, neither tar nor char is considered in the *fuel_comb* stream. Then char is present

and tar not. Followed by the vice versa case and finally both char and tar are assumed to be present in the *fuel_comb* stream.

Code 4.391: Heat loss of the combustion reactor

```
fQ_loss_comb:    Q_loss_comb = P_th_comb*(q_loss_rel_comb/100.0);
```

Code 4.392: Pressure drops for the combustion reactor

```
fdp_air_comb:    air_comb.p = drain_comb.p + dp_air_comb;
fdp_air_rel_comb:        0.01*dp_air_rel_comb = dp_air_comb/
    air_comb.p;
fdp_fuel_comb:   fuel_comb.p = drain_comb.p + dp_fuel_comb;
fdp_fuel_rel_comb: 0.01*dp_fuel_rel_comb = dp_fuel_comb/fuel_comb.
    p;
```

Further, the characteristic values of *Q_loss* (Code 4.391) and several pressure drops (Code 4.392) are calculated.

Code 4.393: Elementary mass balance

```
fAr_1:   drain_comb.massflow*drain_comb.Gas.wAr = air_comb.massflow
    *air_comb.Gas.wAr + fuel_comb.massflow*fuel_comb.Gas.wAr;
fC_1:    drain_comb.massflow*12.011*(2*drain_comb.Gas.wC2H4/28.0536
     + 2*drain_comb.Gas.wC2H6/30.0694 + 3*drain_comb.Gas.wC3H8
    /44.0962 + drain_comb.Gas.wCH4/16.0428 + drain_comb.Gas.wCO
    /28.0104 + drain_comb.Gas.wCO2/44.0098 + drain_comb.Gas.wHCN
    /27.02568) = air_comb.massflow*12.011*(2*air_comb.Gas.wC2H4
    /28.0536 + 2*air_comb.Gas.wC2H6/30.0694 + 3*air_comb.Gas.wC3H8
    /44.0962 + air_comb.Gas.wCH4/16.0428 + air_comb.Gas.wCO/28.0104
     + air_comb.Gas.wCO2/44.0098 + air_comb.Gas.wHCN/27.02568) +
    fuel_comb.massflow*12.011*(2*fuel_comb.Gas.wC2H4/28.0536 + 2*
    fuel_comb.Gas.wC2H6/30.0694 + 3*fuel_comb.Gas.wC3H8/44.0962 +
    fuel_comb.Gas.wCH4/16.0428 + fuel_comb.Gas.wCO/28.0104 +
    fuel_comb.Gas.wCO2/44.0098 + fuel_comb.Gas.wHCN/27.02568);
                              .
                              .
fCl_1:   drain_comb.massflow*35.4527*drain_comb.Gas.wHCl/36.46064 =
     air_comb.massflow*35.4527*air_comb.Gas.wHCl/36.46064 +
    fuel_comb.massflow*35.4527*fuel_comb.Gas.wHCl/36.46064;
endifl
```

As described for the thermal power *P_therm_comb* (Code 4.390) also for the elementary mass balance the four different scenarios are considered. Code 4.393 just shows the first case when neither char nor tar are present in the fuel stream. The mass balance as well as the upcoming balances and assumptions are necessary for the definition of the drain stream composition of the combustion chamber *drain_comb*.

Code 4.394: Inorganic solids section (Dust)

```
ifl !ref(air_comb.Dust) && ref(fuel_comb.Dust) && ref(drain_comb.
    Dust)
        && ref(fuel_comb.Dust) == ref(drain_comb.Dust)
then
```

```
fdust_cont:      fuel_comb.nvolflow*fuel_comb.dust_content =
    drain_comb.nvolflow*drain_comb.dust_content;
endifl

ifl !ref(air_comb.Dust) && ref(fuel_comb.Dust) && ref(drain_comb.
    Dust)
        && ref(fuel_comb.Dust) != ref(drain_comb.Dust)
then
fAsh_comb: fuel_comb.nvolflow*fuel_comb.dust_content*fuel_comb.
    Dust.wAsh = drain_comb.nvolflow*drain_comb.dust_content*
    drain_comb.Dust.wAsh;
fK2O_comb: fuel_comb.nvolflow*fuel_comb.dust_content*fuel_comb.
    Dust.wK2O = drain_comb.nvolflow*drain_comb.dust_content*
    drain_comb.Dust.wK2O;
fMgO_comb: fuel_comb.nvolflow*fuel_comb.dust_content*fuel_comb.
    Dust.wMgO = drain_comb.nvolflow*drain_comb.dust_content*
    drain_comb.Dust.wMgO;
fCaO_comb: fuel_comb.nvolflow*fuel_comb.dust_content*fuel_comb.
    Dust.wCaO = drain_comb.nvolflow*drain_comb.dust_content*
    drain_comb.Dust.wCaO;
fSiO2_comb: fuel_comb.nvolflow*fuel_comb.dust_content*fuel_comb.
    Dust.wSiO2 = drain_comb.nvolflow*drain_comb.dust_content*
    drain_comb.Dust.wSiO2;
fOlivine_comb: fuel_comb.nvolflow*fuel_comb.dust_content*fuel_comb
    .Dust.wOlivine = drain_comb.nvolflow*drain_comb.dust_content*
    drain_comb.Dust.wOlivine;
fCaCO3_comb: fuel_comb.nvolflow*fuel_comb.dust_content*fuel_comb.
    Dust.wCaCO3 = drain_comb.nvolflow*drain_comb.dust_content*
    drain_comb.Dust.wCaCO3;
fDolomite_comb: fuel_comb.nvolflow*fuel_comb.dust_content*
    fuel_comb.Dust.wDolomite = drain_comb.nvolflow*drain_comb.
    dust_content*drain_comb.Dust.wDolomite;
fCaSO4_comb: fuel_comb.nvolflow*fuel_comb.dust_content*fuel_comb.
    Dust.wCaSO4 = drain_comb.nvolflow*drain_comb.dust_content*
    drain_comb.Dust.wCaSO4;
fCaOH2_comb: fuel_comb.nvolflow*fuel_comb.dust_content*fuel_comb.
    Dust.wCaOH2 = drain_comb.nvolflow*drain_comb.dust_content*
    drain_comb.Dust.wCaOH2;
endifl
```

Code 4.394 is responsible for the composition of the share of inorganic solids of stream *drain_comb*. Two different cases are considered, the upper if-condition is applied when the fuel and drain stream of the combustor are using the same dust global. If not, the second case is applied. As for the determination of all the other inorganic solids compositions, the drain dust content is determined by the set of elementary balances (Code 4.394) together with Code 4.57, which applies the sum of all inorganic mass fractions to be one.

Code 4.395: Air ratio

```
flambda_comb:    lambda_comb*drain_comb.massflow*(drain_comb.Gas.
    wO2 - drain_comb.Gas.wCO*31.9988/56.0208) = (air_comb.massflow*
    air_comb.Gas.wO2 + fuel_comb.massflow*fuel_comb.Gas.wO2)*(
    lambda_comb - 1.0);
```

Code 4.396: No gaseous organic carbon in flue gas

```
frC2H4: drain_comb.Gas.yC2H4 = 0.0;
frC2H6: drain_comb.Gas.yC2H6 = 0.0;
frC3H8: drain_comb.Gas.yC3H8 = 0.0;
frCH4:  drain_comb.Gas.yCH4 = 0.0;
```

Code 4.397: CO slip

```
frCO:   if lambda_comb >= 1.0   then
drain_comb.Gas.yCO = (CO_slip_comb/100.0)*drain_comb.Gas.yCO2;
else drain_comb.Gas.yO2 = 0.5*(CO_slip_comb/100.0)*drain_comb.Gas.
    yCO2;
```

Code 4.398: No hydrogen, hydrogen sulfide, hydrogen cyanide, nitrous oxide, or ammonia in drain gas

```
frH2:   drain_comb.Gas.yH2 = 0.0;
frH2S:  drain_comb.Gas.yH2S = 0.0;
frHCN:  drain_comb.Gas.yHCN = 0.0;
frN2O:  drain_comb.Gas.yN2O = 0.0;
frNH3:  drain_comb.Gas.yNH3 = 0.0;
```

Code 4.399: No thermal NO is formed

```
frN2:    drain_comb.nvolflow*drain_comb.Gas.yN2 = air_comb.
    nvolflow*air_comb.Gas.yN2 + fuel_comb.nvolflow*fuel_comb.Gas.
    yN2;
```

The same set of equations from Code 4.395 to 4.399 is used and described for the gas oxidation reactor (Code 4.370 to 4.375).

Pyrolysis reactor

Code 4.400: Energy balance of the pyrolysis reactor

```
fh_total:       drain_gas.massflow*drain_gas.h_total + drain_char.
    massflow*drain_char.h_total = Q_trans*3600.0 + feed_gas.
    massflow*feed_gas.h_total + feed_org.massflow*feed_org.h_total;
```

Code 4.401: Fuel power of the pyrolysis reactor

```
fP_th_pyr: P_th_pyr*3600.0 = feed_org.massflow*feed_org.lhv;
```

Code 4.402: Pressure drops

```
fdp_feed_org:   feed_org.p = drain_gas.p + dp_feed_org;
fdp_feed_org_rel:       0.01*dp_feed_org_rel = dp_feed_org/
    feed_org.p;
fdp_feed_gas:   feed_gas.p = drain_gas.p + dp_feed_gas;
fdp_feed_gas_rel:       0.01*dp_feed_gas_rel = dp_feed_gas/
    feed_gas.p;
fdp_drain_gas_char: drain_gas.p = drain_char.p + dp_drain_gas_char
    ;
```

Code 4.403: Temperature difference in drain_gas and drain_char

```
fdt_drain_gas_char:      drain_gas.t = drain_char.t +
    dt_drain_gas_char;
```

The characteristics values as *Q_trans*, the fuel power -which is in case of pyrolysis defined as the chemical power of the organic feed-, pressure drops and the temperature difference between the *drain_gas* and *drain_char* streams are calculated by Code 4.400 to 4.403.

Code 4.404: Elementary mass balances for the pyrolysis reactor

```
fAr: drain_gas.massflow*drain_gas.Gas.wAr = feed_gas.massflow*
    feed_gas.Gas.wAr;
fC: drain_gas.massflow*12.011*(2*drain_gas.Gas.wC2H4/28.0536 + 2*
    drain_gas.Gas.wC2H6/30.0694 + 3*drain_gas.Gas.wC3H8/44.0962 +
    drain_gas.Gas.wCH4/16.0428 + drain_gas.Gas.wCO/28.0104 +
    drain_gas.Gas.wCO2/44.0098 + drain_gas.Gas.wHCN/27.02568) +
    drain_gas.nvolflow*((drain_gas.char_content/1000.0)*drain_gas.
    Char.wC + (drain_gas.tar_content/1000.0)*drain_gas.Tar.wC) +
    drain_char.massflow*(1.0 - drain_char.water_content -
    drain_char.ash_content)*drain_char.Organic.wC = feed_org.
    massflow*(1.0 - feed_org.water_content - feed_org.ash_content)*
    feed_org.Organic.wC + feed_gas.massflow*12.011*(2*feed_gas.Gas.
    wC2H4/28.0536 + 2*feed_gas.Gas.wC2H6/30.0694 + 3*feed_gas.Gas.
    wC3H8/44.0962 + feed_gas.Gas.wCH4/16.0428 + feed_gas.Gas.wCO
    /28.0104 + feed_gas.Gas.wCO2/44.0098 + feed_gas.Gas.wHCN
    /27.02568);
                              .
                              .
fCl: drain_gas.massflow*35.4527*drain_gas.Gas.wHCl/36.46064 +
    drain_gas.nvolflow*((drain_gas.char_content/1000.0)*drain_gas.
    Char.wCl + (drain_gas.tar_content/1000.0)*drain_gas.Tar.wCl) +
    drain_char.massflow*(1.0 - drain_char.water_content -
    drain_char.ash_content)*drain_char.Organic.wCl = feed_org.
    massflow*(1.0 - feed_org.water_content - feed_org.ash_content)*
    feed_org.Organic.wCl + feed_gas.massflow*35.4527*feed_gas.Gas.
    wHCl/36.46064;
```

The elementary mass balance of Code 4.404 influences the gas and organic drain stream compositions.

Code 4.405: Inorganic species balances

```
fAsh:    drain_gas.nvolflow*(drain_gas.dust_content/1000.0)*
    drain_gas.Dust.wAsh + drain_char.massflow*drain_char.
    ash_content*drain_char.Ash.wAsh = feed_org.massflow*feed_org.
    ash_content*feed_org.Ash.wAsh;
fK2O:    drain_gas.nvolflow*(drain_gas.dust_content/1000.0)*
    drain_gas.Dust.wK2O + drain_char.massflow*drain_char.
    ash_content*drain_char.Ash.wK2O = feed_org.massflow*feed_org.
    ash_content*feed_org.Ash.wK2O;
                              .
                              .
fCaOH2: drain_gas.nvolflow*(drain_gas.dust_content/1000.0)*
    drain_gas.Dust.wCaOH2 + drain_char.massflow*drain_char.
```

```
    ash_content*drain_char.Ash.wCaOH2 = feed_org.massflow*feed_org.
    ash_content*feed_org.Ash.wCaOH2;
```

All inorganic species are treated as chemically invariant, see Code 4.404.

Code 4.406: Splitting conditions for organic substances

```
ffract_char_bed: drain_char.massflow*(1.0 - drain_char.
    water_content - drain_char.ash_content) = feed_org.massflow
    *(1.0 - feed_org.water_content - feed_org.ash_content)*
    fract_char_bed;
ffract_char_pg: drain_gas.nvolflow*(drain_gas.char_content/1000.0)
     = feed_org.massflow*(1.0 - feed_org.water_content - feed_org.
    ash_content)*fract_char_pg;
ffract_tar_pg:  drain_gas.nvolflow*(drain_gas.tar_content/1000.0)
    = feed_org.massflow*(1.0 - feed_org.water_content - feed_org.
    ash_content)*fract_tar_pg;
fungsfd_C: drain_char.massflow*(1.0 - drain_char.water_content -
    drain_char.ash_content)*drain_char.Organic.wC + drain_gas.
    nvolflow*(drain_gas.char_content/1000.0)*drain_gas.Char.wC =
    feed_org.massflow*(1.0 - feed_org.water_content - feed_org.
    ash_content)*feed_org.Organic.wC*unconv_C/100.0;
```

The composition of the exiting organic substances as well as the contents of char and tar in the exit streams have to be set. Otherwise Code 4.406 will not work.

Code 4.407: Splitting conditions for inorganic substances

```
fdust_Ash: drain_gas.nvolflow*(drain_gas.dust_content/1000.0)*
    drain_gas.Dust.wAsh        = feed_org.massflow*feed_org.
    ash_content*fract_fly_ash*feed_org.Ash.wAsh;
fdust_K2O: drain_gas.nvolflow*(drain_gas.dust_content/1000.0)*
    drain_gas.Dust.wK2O        = feed_org.massflow*feed_org.
    ash_content*fract_fly_ash*feed_org.Ash.wK2O;
fdust_MgO: drain_gas.nvolflow*(drain_gas.dust_content/1000.0)*
    drain_gas.Dust.wMgO        = feed_org.massflow*feed_org.
    ash_content*fract_fly_ash*feed_org.Ash.wMgO;
fdust_CaO: drain_gas.nvolflow*(drain_gas.dust_content/1000.0)*
    drain_gas.Dust.wCaO        = feed_org.massflow*feed_org.
    ash_content*fract_fly_ash*feed_org.Ash.wCaO;
fdust_SiO2: drain_gas.nvolflow*(drain_gas.dust_content/1000.0)*
    drain_gas.Dust.wSiO2       = feed_org.massflow*feed_org.
    ash_content*fract_fly_ash*feed_org.Ash.wSiO2;
fdust_Olivine: drain_gas.nvolflow*(drain_gas.dust_content/1000.0)*
    drain_gas.Dust.wOlivine        = feed_org.massflow*feed_org.
    ash_content*fract_fly_ash*feed_org.Ash.wOlivine;
fdust_CaCO3: drain_gas.nvolflow*(drain_gas.dust_content/1000.0)*
    drain_gas.Dust.wCaCO3    = feed_org.massflow*feed_org.
    ash_content*fract_fly_ash*feed_org.Ash.wCaCO3;
fdust_Dolomite: drain_gas.nvolflow*(drain_gas.dust_content/1000.0)
    *drain_gas.Dust.wDolomite      = feed_org.massflow*feed_org.
    ash_content*fract_fly_ash*feed_org.Ash.wDolomite;
fdust_CaSO4: drain_gas.nvolflow*(drain_gas.dust_content/1000.0)*
    drain_gas.Dust.wCaSO4    = feed_org.massflow*feed_org.
    ash_content*fract_fly_ash*feed_org.Ash.wCaSO4;
```

```
fdust_CaOH2: drain_gas.nvolflow*(drain_gas.dust_content/1000.0)*
    drain_gas.Dust.wCaOH2    = feed_org.massflow*feed_org.
    ash_content*fract_fly_ash*feed_org.Ash.wCaOH2;
```

The inorganic substances of bed material and fuel ash do not react chemically in the pyrolyser. Therefore, the contents and compositions of the inorganic solids -exiting the pyrolyser through char and dust in producer gas- can be calculated according to the occurring splitting conditions. In Code 4.407 the dust content of the producer gas as well as its dust composition are expressed as a function of the entering streams and splitting conditions. The inorganic species balances of Code 4.405 yields to the composition and mass flow of the bed material *drain_gas*.

Global quantities

Code 4.408: Exergy loss

```
fE_loss:          drain_comb.Exergy + drain_gas.Exergy + drain_char.
    Exergy + E_loss = air_comb.Exergy + fuel_comb.Exergy + feed_org
    .Exergy + feed_gas.Exergy;
```

The exergy loss is relevant for both, combustion and pyrolysis reactor.

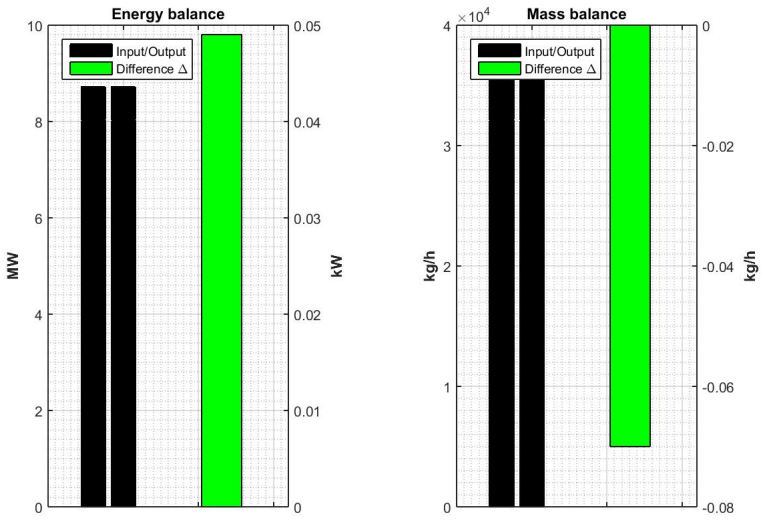


Figure 4.69: Energy and mass balance for the externally heated reactor

Figure 4.69 shows the energy and mass balance of the pyrolyzer unit.

Water quench u_quench_w_

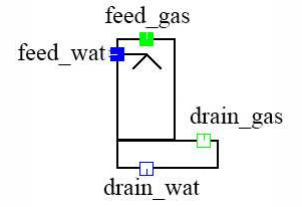


Figure 4.70: Quench library icon

A water quench cools a gas stream via partly evapourating water. The liquid phase is fed through the inlet *feed_wat* and exits through *drain_wat*. The gas stream enters at the terminal *feed_gas* and leaves at *drain_gas*.

Variables	Description
dp_gas	Pressure drop for gas stream [bar]
dp_wat	Pressure drop for water stream [bar]
dp_drain	Pressure difference between water and gas stream exiting the quench [bar]
dt_drain	Temperature difference between exiting gas and exiting water [°C]
E_loss	Exergy loss [kW]

Table 4.49: Quench model item description

Code 4.409: Energy balance

```
fEnergy:        drain_gas.massflow*drain_gas.h_total + drain_wat.
    massflow*drain_wat.h_total = feed_gas.massflow*feed_gas.h_total
     + feed_wat.massflow*feed_wat.h_total;
```

Code 4.410: Water mass balance

```
fmass_H2O:      drain_gas.massflow*drain_gas.Gas.wH2O + drain_wat.
    massflow = feed_gas.massflow*feed_gas.Gas.wH2O + feed_wat.
    massflow;
```

Code 4.411: Mass balance gaseous components

```
fmass_Ar:       drain_gas.massflow*drain_gas.Gas.wAr = feed_gas.
    massflow*feed_gas.Gas.wAr;
fmass_C2H4:     drain_gas.massflow*drain_gas.Gas.wC2H4 = feed_gas.
    massflow*feed_gas.Gas.wC2H4;
fmass_C2H6:     drain_gas.massflow*drain_gas.Gas.wC2H6 = feed_gas.
    massflow*feed_gas.Gas.wC2H6;
fmass_C3H8:     drain_gas.massflow*drain_gas.Gas.wC3H8 = feed_gas.
    massflow*feed_gas.Gas.wC3H8;
fmass_CH4:      drain_gas.massflow*drain_gas.Gas.wCH4 = feed_gas.
    massflow*feed_gas.Gas.wCH4;
```

```
fmass_CO:       drain_gas.massflow*drain_gas.Gas.wCO = feed_gas.
    massflow*feed_gas.Gas.wCO;
fmass_CO2:      drain_gas.massflow*drain_gas.Gas.wCO2 = feed_gas.
    massflow*feed_gas.Gas.wCO2;
fmass_H2:       drain_gas.massflow*drain_gas.Gas.wH2 = feed_gas.
    massflow*feed_gas.Gas.wH2;
fmass_H2S:      drain_gas.massflow*drain_gas.Gas.wH2S = feed_gas.
    massflow*feed_gas.Gas.wH2S;
fmass_HCl:      drain_gas.massflow*drain_gas.Gas.wHCl = feed_gas.
    massflow*feed_gas.Gas.wHCl;
fmass_HCN:      drain_gas.massflow*drain_gas.Gas.wHCN = feed_gas.
    massflow*feed_gas.Gas.wHCN;
fmass_N2:       drain_gas.massflow*drain_gas.Gas.wN2 = feed_gas.
    massflow*feed_gas.Gas.wN2;
fmass_N2O:      drain_gas.massflow*drain_gas.Gas.wN2O = feed_gas.
    massflow*feed_gas.Gas.wN2O;
fmass_NH3:      drain_gas.massflow*drain_gas.Gas.wNH3 = feed_gas.
    massflow*feed_gas.Gas.wNH3;
fmass_NO:       drain_gas.massflow*drain_gas.Gas.wNO = feed_gas.
    massflow*feed_gas.Gas.wNO;
fmass_O2:       drain_gas.massflow*drain_gas.Gas.wO2 = feed_gas.
    massflow*feed_gas.Gas.wO2;
fmass_SO2:      drain_gas.massflow*drain_gas.Gas.wSO2 = feed_gas.
    massflow*feed_gas.Gas.wSO2;
```

The model's energy balance is defined by Code 4.409. The mass balance for water (Code 4.410) considers water in gaseous phase as well as in its liquid form. The species mass balances of the remaining gas components is defined by Code 4.411.

Code 4.412: Mass balance dust

```
if1 ref(drain_gas.Dust) then
fmass_dust: drain_gas.nvolflow*drain_gas.dust_content = feed_gas.
    nvolflow*feed_gas.dust_content;
endif1
```

Code 4.413: Mass balance char

```
if1 ref(drain_gas.Char) then
fmass_char:     drain_gas.nvolflow*drain_gas.char_content =
    feed_gas.nvolflow*feed_gas.char_content;
endif1
```

Code 4.414: Mass balance tar

```
if1 ref(drain_gas.Char) then
fmass_char:     drain_gas.nvolflow*drain_gas.char_content =
    feed_gas.nvolflow*feed_gas.char_content;
endif1
```

The mass balances for dust, char and tar are writen in Code 4.412 to 4.414.

Code 4.415: Pressure drops

```
fdp_gas:        drain_gas.p + dp_gas = feed_gas.p;
fdp_wat:        drain_wat.p + dp_wat = feed_wat.p;
```

Code 4.416: Pressure difference between exiting gas and exiting water

```
fdp_drain:      drain_gas.p + dp_drain = drain_wat.p;
```

Code 4.417: Temperature difference between exiting gas and exiting water

```
fdt_drain:      drain_gas.t = drain_wat.t + dt_drain;
```

Code 4.418: Exergy loss

```
fE_loss: drain_gas.Exergy + drain_wat.Exergy + E_loss = feed_gas.
    Exergy + feed_wat.Exergy;
```

Common unit characteristics of Table 4.49 are defined by Code 4.415 to 4.418.

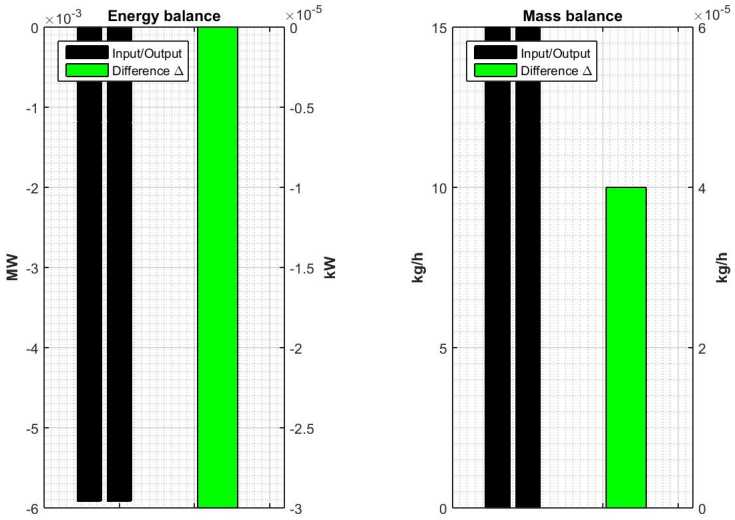


Figure 4.71: Energy and mass balance for the quench model

Also for this unit a energy and mass balance is included the proof the correctness of its results (Fig. 4.71).

Saturation temperature and pressure calculation u_sat_tp_

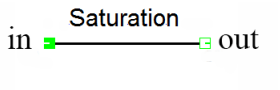


Figure 4.72: Saturation library icon

The gas stream enters at the terminal *feed_gas* and leaves at *drain_gas*. Through out the balances from Code 4.419 and 4.420 it is defined that the stream does not change in any way while passing this unit. From Code 4.421 to 4.423 the saturation conditions for the gas stream are calculated.

Variables	Description
T	Temperature of the gas stream [°C]
T_sat	Saturation temperature of the gas [°C]
dT_sat	Difference between the saturation temperature to the actual gas temperature [°C]
p	Pressure of the gas stream [bar]
p_sat	Pressure at which the water fraction of the gas reaches its saturation pressure [bar]
dp_sat	Difference between actual gas stream pressure and the saturation pressure for water condensation [bar]
phi_rel	Relative humidity of the gas stream [%]
Function	**Description**
wfp0t	Function returns vapor pressure of water at a certain temperature

Table 4.50: Saturation model item description

Code 4.419: Mass and energy balance

```
f1:     in.p = out.p;
f2:     in.massflow = out.massflow;
f3:     in.t = out.t;
```

The balance of mass and energy are summarized in Code 4.419. Instead of a typical energy balance it is defined that pressure, massflow and temperature do not change.

Code 4.420: Dust, char and tar balance

```
if1 ref(in.Dust) || ref(out.Dust) then
f1_dust:        in.dust_content = out.dust_content;
endif1
if1 ref(in.Char) || ref(out.Char) then
f1_char:        in.char_content = out.char_content;
endif1
if1 ref(in.Tar) || ref(out.Tar) then
f1_tar: in.tar_content = out.tar_content;
endif1
```

In Code 4.419 the mass balances for dust, char and tar are fixed.

Code 4.421: Calculation of the saturation temperature

```
fT_sat: in.p * in.Gas.yH2O =    wfp0t(T_sat);
fdT_sat:          dT_sat = in.t - T_sat;
fT:       T = in.t;
```

Code 4.422: Calculation of the saturation pressure

```
fp_sat: p_sat = wfp0t(T)/in.Gas.yH2O;
fdp_sat:          dp_sat =  p_sat - in.p;
fp:       p = in.p;
```

Code 4.423: Calculation of the relative humidity of the gas stream

```
f_phi_rel:        phi_rel = p / p_sat * 100.0;
```

Through the last three code sections the saturation conditions are calculated.

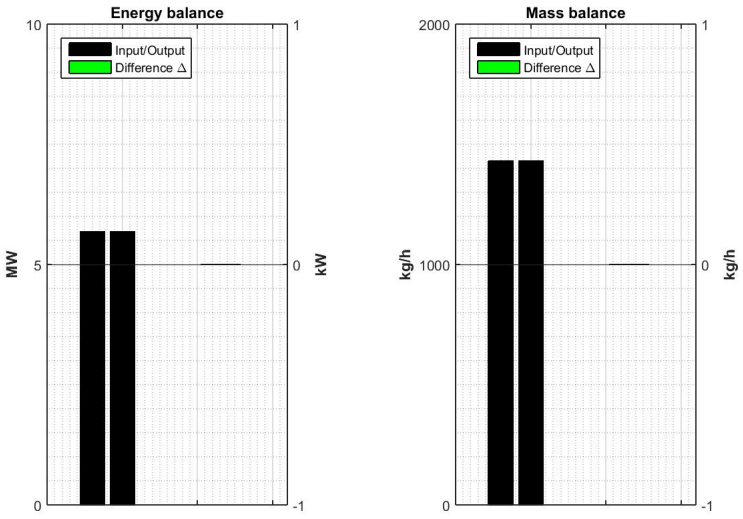


Figure 4.73: Energy and mass balance for the gas saturation model

The units correct operation is shown through the energy and mass balance of this unit in Figure 4.73.

Organic scrubber u_scrub_org_

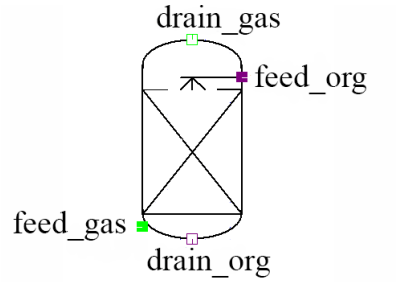


Figure 4.74: Organic scrubber library icon

The organic scrubber model absorbs organic tar from a gas streams using organic solvents. As a side effect, water is condensing. The gas stream is fed through the terminal *feed_gas* and leaves at *drain _gas* while the organic inlet is called *feed_org* and the exit *drain_org*. If the gas stream contains dust or char, it is removed by the solvent to a defined degree. Dust, char, and tar definitions are required in the feed gas. In the drain gas stream the definition of the organic tar global is compulsory, the dust and char globals can be referenced optionally. The density of the organic scrubbing agent has to be set either in the feed or drain stream. Useful theoretical information on the scrubber unit can be found in [76][5.1.6].

Variables	Description
dp_gas	Gas pressure drop in the scrubber [bar]
dp_org	Pressure drop for the organic solvent in the scrubber [bar]
dt_top	Temperature difference between gas drain and solvent feed [°C]
sep_eff_dust	Separation efficiency for the dust [%]
sep_eff_char	Separation efficiency for the char [%]
sep_eff_tar	Separation efficiency for tar [%]
Q_trans_solv	Power which has to be removed by the solvent [kW]
q_loss_rel	Heat loss of the scrubber in relation to the sensible enthalpy of the gas feed [%]
Q_loss	Heat loss of the scrubber [kW]
E_loss	Exergy loss of the scrubber [kW]
ratio_solv_gas	Ratio between mass flow of incoming organic solvent and mass flow of total incoming gas [-]
Function	**Description**
wfp0t	Function returns vapour pressure of water at a certain temperature [°C]

Table 4.51: Organic scrubber model item description

Code 4.424: Pressure drops

```
fdp_gas:        drain_gas.p + dp_gas = feed_gas.p;
fdp_org:        drain_org.p + dp_org = feed_org.p;
```

Code 4.425: Temperature difference between gas drain and solvent feed

```
fdt_top: drain_gas.t = feed_org.t + dt_top;
```

Code 4.426: Density equality between solvent feed and solvent drain

```
forg_dens: (1/feed_org.rho - feed_org.ash_content/feed_org.Ash.rho
    - feed_org.water_content/1000.0) / (1-feed_org.water_content-
    feed_org.ash_content) = (1/drain_org.rho - drain_org.
    ash_content/drain_org.Ash.rho - drain_org.water_content/1000.0)
    / (1-drain_org.water_content-drain_org.ash_content);
```

Code 4.427: Energy balance

```
fh_total: drain_org.massflow*drain_org.h_total + drain_gas.
    massflow*drain_gas.h_total + Q_loss*3600.0 = feed_gas.massflow*
    feed_gas.h_total + feed_org.massflow*feed_org.h_total;
```

Code 4.428: Heat loss

```
fQ_loss: Q_loss*3600.0 = (feed_gas.massflow*feed_gas.h + (feed_gas
    .nvolflow/1000)*(feed_gas.dust_content*feed_gas.h_dust +
    feed_gas.char_content*feed_gas.h_char + feed_gas.tar_content*
    feed_gas.h_tar))*q_loss_rel/100;
```

Code 4.429: Heat absorbed by the solvent

```
fQ_trans_solv: drain_org.massflow*((1.0 - drain_org.water_content
    - drain_org.ash_content)*drain_org.h_waf + drain_org.
    water_content*drain_org.wfhpt(drain_org.p, drain_org.t) +
    drain_org.ash_content*drain_org.Ash.sfht(drain_org.t)) =
    Q_trans_solv*3600.0 + feed_org.massflow*((1.0 - feed_org.
    water_content - feed_org.ash_content)*feed_org.h_waf + feed_org
    .water_content*feed_org.wfhpt(feed_org.p, feed_org.t) +
    feed_org.ash_content*feed_org.Ash.sfht(feed_org.t));
```

Characteristic values of Table 4.51 are defined and calculated via Code 4.424 to 4.429.

Code 4.430: Elementary mass balances

```
fHchar: drain_gas.massflow*(4*drain_gas.Gas.wC2H4*1.00794/28.05376
    + 6*drain_gas.Gas.wC2H6*1.00794/30.06964 + 8*drain_gas.Gas.
    wC3H8*1.00794/44.09652 + 4*drain_gas.Gas.wCH4*1.00794/16.04276
    + drain_gas.Gas.wH2 + 2*drain_gas.Gas.wH2O*1.00794/18.01528 +
    2*drain_gas.Gas.wH2S*1.00794/34.08188 + drain_gas.Gas.wHCl
    *1.00794/36.46064 + drain_gas.Gas.wHCN*1.00794/27.02568 + 3*
    drain_gas.Gas.wNH3*1.00794/17.03056) + drain_org.massflow*((1.0
    - drain_org.water_content - drain_org.ash_content)*drain_org.
    Organic.wH + 2*drain_org.water_content*1.00794/18.01528) + (
    drain_gas.nvolflow/1000)*(drain_gas.char_content*drain_gas.Char
    .wH + drain_gas.tar_content*drain_gas.Tar.wH) = feed_gas.
    massflow*(4*feed_gas.Gas.wC2H4*1.00794/28.0536 + 6*feed_gas.Gas
    .wC2H6*1.00794/30.0694 + 8*feed_gas.Gas.wC3H8*1.00794/44.0962 +
    4*feed_gas.Gas.wCH4*1.00794/16.0428 + feed_gas.Gas.wH2 + 2*
```

```
    feed_gas.Gas.wH2O*1.00794/18.01528 + 2*feed_gas.Gas.wH2S
    *1.00794/34.08188 + feed_gas.Gas.wHCl*1.00794/36.46064 +
    feed_gas.Gas.wHCN*1.00794/27.02568 + 3*feed_gas.Gas.wNH3
    *1.00794/17.03056) + (feed_gas.nvolflow/1000)*(feed_gas.
    char_content*feed_gas.Char.wH + feed_gas.tar_content*feed_gas.
    Tar.wH) + feed_org.massflow*((1.0 - feed_org.water_content -
    feed_org.ash_content)*feed_org.Organic.wH + 2*feed_org.
    water_content*1.00794/18.01528);
                              .
                              .
fClchar: drain_gas.massflow*drain_gas.Gas.wHCl*35.4527/36.46064 +
    drain_org.massflow*(1.0 - drain_org.water_content - drain_org.
    ash_content)*drain_org.Organic.wCl + (drain_gas.nvolflow/1000)
    *(drain_gas.char_content*drain_gas.Char.wCl + drain_gas.
    tar_content*drain_gas.Tar.wCl) = feed_gas.massflow*feed_gas.Gas
    .wHCl*35.4527/36.46064 + (feed_gas.nvolflow/1000)*(feed_gas.
    char_content*feed_gas.Char.wCl + feed_gas.tar_content*feed_gas.
    Tar.wCl) + feed_org.massflow*(1.0 - feed_org.water_content -
    feed_org.ash_content)*feed_org.Organic.wCl;
elsel
fH:      drain_gas.massflow*(4*drain_gas.Gas.wC2H4*1.00794/28.05376
     + 6*drain_gas.Gas.wC2H6*1.00794/30.06964 + 8*drain_gas.Gas.
    wC3H8*1.00794/44.09652 + 4*drain_gas.Gas.wCH4*1.00794/16.04276
    + drain_gas.Gas.wH2 + 2*drain_gas.Gas.wH2O*1.00794/18.01528 +
    2*drain_gas.Gas.wH2S*1.00794/34.08188 + drain_gas.Gas.wHCl
    *1.00794/36.46064 + drain_gas.Gas.wHCN*1.00794/27.02568 + 3*
    drain_gas.Gas.wNH3*1.00794/17.03056) + drain_gas.nvolflow*(
    drain_gas.tar_content/1000)*drain_gas.Tar.wH + drain_org.
    massflow*((1.0 - drain_org.water_content - drain_org.
    ash_content)*drain_org.Organic.wH + 2*drain_org.water_content
    *1.00794/18.01528) = feed_gas.massflow*(4*feed_gas.Gas.wC2H4
    *1.00794/28.0536 + 6*feed_gas.Gas.wC2H6*1.00794/30.0694 + 8*
    feed_gas.Gas.wC3H8*1.00794/44.0962 + 4*feed_gas.Gas.wCH4
    *1.00794/16.0428 + feed_gas.Gas.wH2 + 2*feed_gas.Gas.wH2O
    *1.00794/18.01528 + 2*feed_gas.Gas.wH2S*1.00794/34.08188 +
    feed_gas.Gas.wHCl*1.00794/36.46064 + feed_gas.Gas.wHCN
    *1.00794/27.02568 + 3*feed_gas.Gas.wNH3*1.00794/17.03056) + (
    feed_gas.nvolflow/1000)*(feed_gas.char_content*feed_gas.Char.wH
     + feed_gas.tar_content*feed_gas.Tar.wH) + feed_org.massflow
    *((1.0 - feed_org.water_content - feed_org.ash_content)*
    feed_org.Organic.wH + 2*feed_org.water_content
    *1.00794/18.01528);
                    .
                    .
fCl: drain_gas.massflow*drain_gas.Gas.wHCl*35.4527/36.46064 +
    drain_gas.nvolflow*(drain_gas.tar_content/1000)*drain_gas.Tar.
    wCl + drain_org.massflow*(1.0 - drain_org.water_content -
    drain_org.ash_content)*drain_org.Organic.wCl = feed_gas.
    massflow*feed_gas.Gas.wHCl*35.4527/36.46064 + (feed_gas.
    nvolflow/1000)*(feed_gas.char_content*feed_gas.Char.wCl +
    feed_gas.tar_content*feed_gas.Tar.wCl)+ feed_org.massflow*(1.0
    - feed_org.water_content - feed_org.ash_content)*feed_org.
    Organic.wCl;
endifl
```

Code 4.430 describes a global elementary balance of the scrubber unit. The first case describes the scenario when char is present in the gaseous feed stream, the second option if not. This code is the beginning of a set of equations to determine the drain gas composition of both stream classes.

Code 4.431: Balances of inorganic components

```
ifl (ref(feed_gas.Dust) != ref(feed_org.Ash)
     && ref(feed_gas.Dust) == ref(drain_org.Ash)) then
fAsh: (drain_gas.nvolflow*(drain_gas.dust_content/1000) +
    drain_org.massflow*drain_org.ash_content)*drain_org.Ash.wAsh =
    feed_gas.nvolflow*(feed_gas.dust_content/1000)*feed_gas.Dust.
    wAsh + feed_org.massflow*feed_org.ash_content*feed_org.Ash.wAsh
    ;
                              .
                              .
fCaOH2: (drain_gas.nvolflow*(drain_gas.dust_content/1000) +
    drain_org.massflow*drain_org.ash_content)*drain_org.Ash.wCaOH2
    = feed_gas.nvolflow*(feed_gas.dust_content/1000)*feed_gas.Dust.
    wCaOH2 + feed_org.massflow*feed_org.ash_content*feed_org.Ash.
    wCaOH2;
endifl
ifl  ref(feed_gas.Dust) == ref(drain_org.Ash)
     &&  ref(feed_gas.Dust) == ref(feed_org.Ash) then
fash_cont:       drain_gas.nvolflow*drain_gas.dust_content +
    drain_org.massflow*drain_org.ash_content*1000.0  = feed_gas.
    nvolflow*feed_gas.dust_content + feed_org.massflow*feed_org.
    ash_content*1000.0;
endifl
```

In Code 4.431, for the first if-condition different inorganic components are considered for the feed streams. For the other if-option it is assumed that the inorganic composition in each one of the four streams is equal and therefore one balance is enough in comparison to the first case.

Code 4.432: Gas composition in gas drain

```
fAr:     drain_gas.massflow*drain_gas.Gas.wAr = feed_gas.massflow*
    feed_gas.Gas.wAr;
fC2H4:   drain_gas.massflow*drain_gas.Gas.wC2H4 = feed_gas.massflow
    *feed_gas.Gas.wC2H4;
                              .
                              .
fSO2:    drain_gas.massflow*drain_gas.Gas.wSO2 = feed_gas.massflow*
    feed_gas.Gas.wSO2;
```

Code 4.433: Water content in gas drain

```
forg_water_cont:          drain_org.massflow*drain_org.water_content
     + drain_gas.massflow*drain_gas.Gas.wH2O = feed_gas.massflow*
    feed_gas.Gas.wH2O + feed_org.massflow*feed_org.water_content;
```

Code 4.434: Balance of water

```
forg_water_cont:          drain_org.massflow*drain_org.water_content
    + drain_gas.massflow*drain_gas.Gas.wH2O = feed_gas.massflow*
    feed_gas.Gas.wH2O + feed_org.massflow*feed_org.water_content;
```

The drain gas composition is mainly defined by Code 4.434. It is assumed that no other gaseous species than water is absorbed by the scrubbing agent. The degree of absorption of water is defined by Code 4.433 and 4.434. Throughout Code 4.433 it is defined that the drain gas water content is at 100% humidity, at its maximum.

Code 4.435: Tar composition in the drain gas stream

```
ifl (ref(feed_gas.Tar) != ref(drain_gas.Tar)) then
ftar_C: feed_gas.Tar.wC = drain_gas.Tar.wC;
ftar_H: feed_gas.Tar.wH = drain_gas.Tar.wH;
ftar_O: feed_gas.Tar.wO = drain_gas.Tar.wO;
ftar_N: feed_gas.Tar.wN = drain_gas.Tar.wN;
ftar_S: feed_gas.Tar.wS = drain_gas.Tar.wS;
endifl
```

Code 4.435 is just applied if the tar global of the feed and drain gas stream differ from each other. If they are different, this code section makes their composition equal even if they are named different.

Code 4.436: Tar separation efficiency

```
fsep_eff_tar:   drain_gas.nvolflow*drain_gas.tar_content = (1.0 -
    sep_eff_tar/100.0)*feed_gas.nvolflow*feed_gas.tar_content;
```

Code 4.437: Char separation efficiency

```
ifl ref(drain_gas.Char) then
fsep_eff_char:   drain_gas.nvolflow*drain_gas.char_content = (1.0 -
    sep_eff_char/100.0)*feed_gas.nvolflow*feed_gas.char_content;
elsel
fsep_eff_char2: sep_eff_char=100;
endifl
```

Code 4.438: Dust separation efficiency

```
ifl ref(drain_gas.Dust) then
fsep_eff_dust:   drain_gas.nvolflow*drain_gas.dust_content = (1.0 -
    sep_eff_dust/100.0)*feed_gas.nvolflow*feed_gas.dust_content;
elsel
fsep_eff_dust2: sep_eff_dust=100;
endifl
```

Code 4.439: Ratio between solvent mass flow and gas mass flow

```
fratio_solv_gas: feed_gas.massflow*ratio_solv_gas = feed_org.
    massflow;
```

Code 4.440: Exergy loss

```
fExergy: drain_gas.Exergy + drain_org.Exergy + E_loss = feed_gas.
    Exergy + feed_org.Exergy;
```

Further characteristic values of Table 4.51 are defined in Code 4.436 to 4.440.

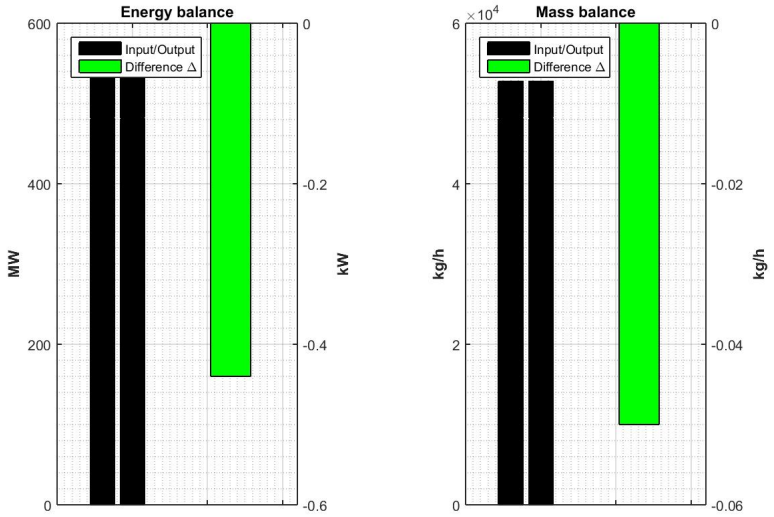


Figure 4.75: Energy and mass balance for the organic scrubber model

Figure 4.75 shows the mass and energy balance of this unit.

Pressure swing adsorption u_sep_g_psa

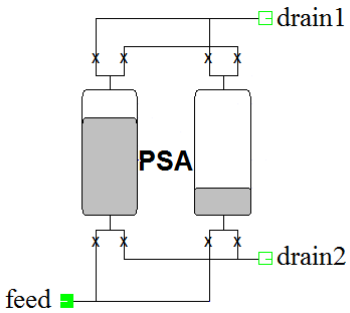


Figure 4.76: Pressure swing adsorption library icon

For the pressure swing adsorption (PSA) model a gas stream is fed through the terminal *feed*. Selected species are adsorbed by an adsorbent within a number of columns at relatively high pressure. Not adsorbed species are passing the column and exit through the

outlet *drain1*. Due to relaxation, the adsorbent is regenerated. The adsorbed and then released species are led away via the connection *drain2*.

Variables	Description
sep_coeff_CO	Separation coefficient considering CO [%]
sep_coeff_H2	Separation coefficient considering H_2 [%]
dp_1	Pressure drop between feed and drain1 connections [bar]
dp_2	Separation efficiency for the dust [bar]
dt_1	Separation efficiency for the char [°C]
dt_2	Separation efficiency for tar [°C]
Q_loss	Heat loss of PSA unit [kW]
E_loss	Exergy loss of PSA unit [kW]

Table 4.52: PSA model item description

Code 4.441: Energy balance

```
fh_total:        drain1.h_total*drain1.massflow + drain2.h_total*
    drain2.massflow + Q_loss*3600.0 = feed.h_total*feed.massflow;
```

Code 4.442: Pressure drops

```
fp1:      feed.p= dp_1 + drain1.p;
fp2:     feed.p = dp_2 + drain2.p;
```

Code 4.443: Temperature drops

```
ft1:      feed.t= dt_1 + drain1.t;
ft2:      feed.t= dt_2 + drain2.t;
```

Code 4.444: Exergy loss

```
f_E_loss:        drain1.Exergy + drain2.Exergy + E_loss = feed.
    Exergy;
```

Characteristic values of Table 4.52 are defined through the equations of Code 4.441 to 4.444.

Code 4.445: Species balance

```
if1       ref(drain1.Gas) != ref(drain2.Gas)
          && ref(drain1.Gas) != ref(feed.Gas)
          && ref(drain2.Gas) != ref(feed.Gas)
then
f1wAr:   feed.massflow * feed.Gas.wAr = drain1.massflow * drain1.
    Gas.wAr + drain2.massflow * drain2.Gas.wAr;
                              .
                              .
f1wSO2: feed.massflow * feed.Gas.wSO2 = drain1.massflow * drain1.
    Gas.wSO2 + drain2.massflow * drain2.Gas.wSO2;
endif1
```

Code 4.445 helps to create the drain gas composition, in particular the composition for the gas global. Code 4.445 is for the general case when all three gas globals (*feed*, *drain1*, *drain2*) are different.

Code 4.446: Dust content

```
if1 ref(drain1.Dust) || ref(drain2.Dust) || ref(feed.Dust) then
fdust_cont:      feed.nvolflow*feed.dust_content = drain1.nvolflow*
    drain1.dust_content + drain2.nvolflow*drain2.dust_content;
endif1
```

Code 4.447: Dust composition

```
if1      ref(drain1.Dust) && ref(drain2.Dust) && ref(feed.Dust)
        && ref(drain1.Dust) != ref(drain2.Dust)
        && ref(drain1.Dust) != ref(feed.Dust)
        && ref(drain2.Dust) != ref(feed.Dust)
then
f11dustwCaCO3: feed.nvolflow*feed.dust_content*feed.Dust.wCaCO3 =
    drain1.nvolflow*drain1.dust_content*drain1.Dust.wCaCO3 + drain2
    .nvolflow*drain2.dust_content*drain2.Dust.wCaCO3;
                              .
                              .
f11dustwOlivine: feed.nvolflow*feed.dust_content*feed.Dust.
    wOlivine      = drain1.nvolflow*drain1.dust_content*drain1.Dust.
    wOlivine + drain2.nvolflow*drain2.dust_content*drain2.Dust.
    wOlivine;
endif1
```

Code 4.446 and 4.447 are responsible for the definition of the dust content and its composition, if all three dust objects are referenced and different. For the composition, the mass balance for ash is not listed as the ash share is calculated by the inorganic solids global (Eq.4.57). Further scenarios considering the dust objects are

- All three Dust objects are referenced and both drain streams use the same dust global, the feed dust is a different object
- All three Dust objects are referenced and the feed uses the same dust global as one of the drain streams, the other drain stream uses a different dust global
- Dust is referenced in *drain1* and *feed*, no Dust referenced in *drain2*
- Dust is referenced in *drain2* and *feed*, no dust referenced in *drain1*

The same case scenarios as for dust are also considered for char and tar.

Code 4.448: Separation coefficients

```
f1sepCO: feed.massflow * feed.Gas.wCO * sep_coef_CO/100 = drain1.
    massflow * drain1.Gas.wCO;
f1sepH2: feed.massflow * feed.Gas.wH2 * sep_coef_H2/100 = drain1.
    massflow * drain1.Gas.wH2;
```

Code 4.449: Definition of composition of drain1 stream

```
fsepAr:     drain1.Gas.wAr   = 0;
fsepC2H4:   drain1.Gas.wC2H4 = 0;
fsepC2H6:   drain1.Gas.wC2H6 = 0;
fsepC3H8:   drain1.Gas.wC3H8 = 0;
fsepCH4:    drain1.Gas.wCH4  = 0;
fsepCO2:    drain1.Gas.wCO2  = 0;
fsepH2O:    drain1.Gas.wH2O  = 0;
fsepH2S:    drain1.Gas.wH2S  = 0;
fsepHCl:    drain1.Gas.wHCl  = 0;
fsepHCN:    drain1.Gas.wHCN  = 0;
fsepN2:     drain1.Gas.wN2   = 0;
fsepN2O:    drain1.Gas.wN2O  = 0;
fsepNH3:    drain1.Gas.wNH3  = 0;
fsepNO:     drain1.Gas.wNO   = 0;
fsepO2:     drain1.Gas.wO2   = 0;
fsepSO2:    drain1.Gas.wSO2  = 0;
```

The composition of *drain1* is defined through Code 4.448 and 4.449. All species of Code 4.449 are not present in stream *drain1*. The hydrogen or carbon monoxide content of stream *drain1* is defined by Code 4.448. The other one -not defined through Code 4.448- is obtained by the gas global, as the sum of all gaseous components equals one (Eq.4.45).

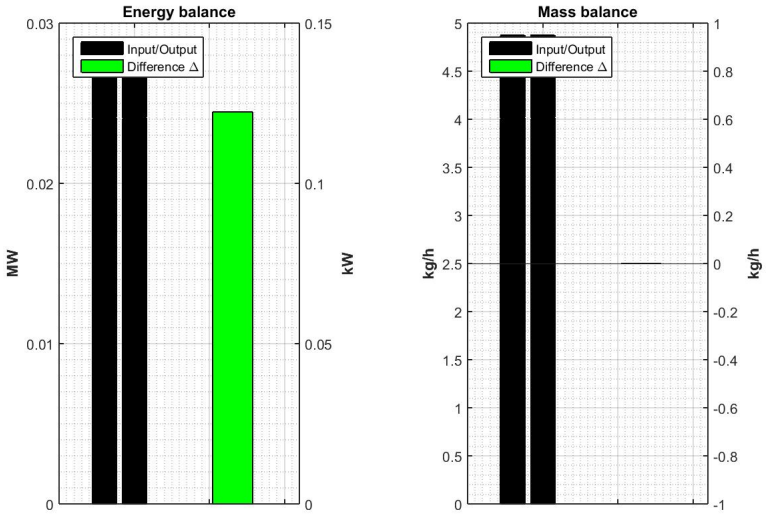


Figure 4.77: Energy and mass balance for the PSA unit

Figure 4.77 presents the PSA's mass energy balance to proof the correctness of its operation.

Separator organic-organic u_sep_o

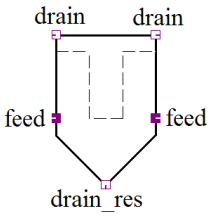


Figure 4.78: Separator organic-organic library icon

The model separates dust and organic compounds from an organic stream according to their separation efficiencies. Dust must be referenced in the organic *feed* stream and in the drain residue stream (*drain_res*). The presence of dust is optional in the organic *drain* stream, if dust is not referenced in the *drain* stream, 100% separation efficiency is assumed. The organic global of the residue stream can be different to the organic global of the *drain* stream, due to the fact that char can also be removed from the organic stream. Additional, theoretical information about this unit can be found in [76][5.1.4].

Variables	Description
dp	Pressure drop [bar]
sep_eff_ash	Separation efficiency for ash [%]
sep_eff_org	Separation efficiency for char [%]
E_loss	Exergy loss of PSA unit [kW]
Function	**Description**
wfp0t	Returns the vapor pressure of water at a certain temperature [°C]

Table 4.53: Separator organic-organic model item description

Code 4.450: Mass balance

```
fmass:    feed.massflow = drain.massflow + drain_res.massflow;
```

Code 4.451: Density balance for the filter

```
forg_dens: (1/feed.rho - feed.ash_content/feed.Ash.rho - feed.
    water_content/1000.0) / (1-feed.water_content-feed.ash_content)
     = (1/drain.rho - drain.ash_content/drain.Ash.rho - drain.
    water_content/1000.0) / (1-drain.water_content-drain.
    ash_content) + (1/drain_res.rho - drain_res.ash_content/
    drain_res.Ash.rho - drain_res.water_content/1000.0) / (1-
    drain_res.water_content-drain_res.ash_content) ;
```

The mass continuity is guaranteed by Code 4.450. As the filter is about an organic stream, the density for the *drain* or *drain_res* stream has to be calculated as well. One of the two mentioned densities has to be set. Code 4.451 calculates the unknown density.

Code 4.452: Pressure drop in the organic filtered stream

```
fdp:    drain.p + dp = feed.p;
```

Code 4.453: Temperature stays the same during filtering

```
ft:     feed.t=drain.t;
ft2:    feed.t =drain_res.t;
```

Code 4.452 and 4.453 calcualte variables from Table 4.53.

Code 4.454: Organic drain composition

```
ifl ref(drain.Organic) != ref(feed.Organic) then
fH:     drain.Organic.wH = feed.Organic.wH;
fN:     drain.Organic.wN = feed.Organic.wN;
fO:     drain.Organic.wO = feed.Organic.wO;
fS:     drain.Organic.wS = feed.Organic.wS;
fCl:    drain.Organic.wCl = feed.Organic.wCl;
endifl
```

If different globals are used for the *feed* and the organic *drain* stream their composition is set equal via Code 4.454.

Code 4.455: Ash balance

```
ifl
        ref (feed.Ash) == ref (drain.Ash)  then
fash_cont:drain.massflow * drain.ash_content*1000.0 + drain_res.
    massflow*drain_res.ash_content*1000.0  = feed.massflow*feed.
    ash_content*1000.0;
endifl
```

The balance of Code 4.455 is applied if the same ash global is used for the *feed* and *drain* stream.

Code 4.456: Balance for water

```
forg_water_cont: drain.massflow*drain.water_content + drain_res.
    massflow*drain_res.water_content = feed.massflow*feed.
    water_content;
```

The water content of the *drain* stream is defined by Code 4.456.

Code 4.457: Separation efficiencies

```
ifl ref(drain.Ash) then
fsep_eff_ash: drain.massflow*drain.ash_content = (1.0 -
    sep_eff_ash/100.0)*feed.massflow*feed.ash_content;
elsel
fsep_eff_ash2:   sep_eff_ash=100.0;
endifl
fsep_eff_org: drain.massflow *(1-drain.water_content-drain.
    ash_content) = (1.0 - sep_eff_org/100.0)*feed.massflow*(1-feed.
    water_content-feed.ash_content) ;
```

Code 4.458: Exergy loss

```
fExergy:        drain.Exergy + drain_res.Exergy + E_loss = feed.
    Exergy;
```

Due to Code 4.457 and 4.458, variables of Table 4.53 are defined.

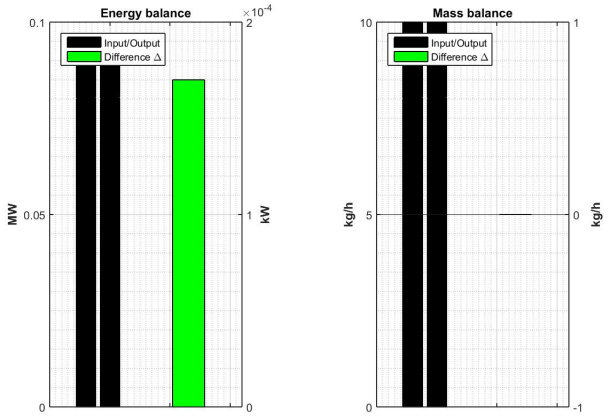


Figure 4.79: Energy and mass balance for the organic-organic separator unit

Figure 4.79 (energy and mass balance) is the proof for the correct operation of the separator unit.

Separator inorganic-organic u_sep_os_

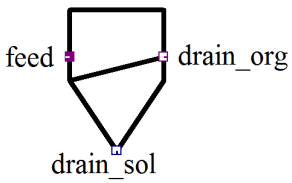


Figure 4.80: Separator inorganic-organic library icon

The model separates inorganic solids from an ash containing organic stream. Ash must be referenced in the organic *feed* and is optionally possible in the organic drain stream (*drain_org*). The solid, inorganic components leave the filter at terminal *drain_sol*. Further expainatory information can be found in [76][5.1.4].

Variables	Description
dp_org	Pressure drop of the organic liquid passing the separator [bar]
dt_drain	Temperature difference between drain streams [°C]
sep_eff_inorg	Separation efficiency of inorganic substance towards the solid drain stream [%]
sep_eff_org	Separation efficiency of organic substance towards the Char fraction in the solid drain stream [%]
E_loss	Exergy loss [kW]

Table 4.54: Separator inorganic-organic model item description

Code 4.459: Energy balance

```
fEnergy:        drain_sol.massflow*drain_sol.h_total + drain_org.
    massflow*drain_org.h_total = feed.massflow*feed.h_total;
```

Code 4.460: Mass balances

```
fmass_org:      drain_org.massflow*(1.0 - drain_org.water_content
    - drain_org.ash_content) + drain_sol.massflow*drain_sol.
    char_content = feed.massflow*(1.0 - feed.water_content - feed.
    ash_content);
fmass_ash:      drain_org.massflow*drain_org.ash_content +
    drain_sol.massflow = feed.massflow*feed.ash_content;
fmass_wat:      drain_org.massflow*drain_org.water_content = feed.
    massflow*feed.water_content;
```

To ensure the correctness of energy and mass balance, Code 4.459 and 4.460 are applied.

Code 4.461: Pressure drops

```
fdp_org:        drain_org.p + dp_org = feed.p;
```

Code 4.462: Temperature difference between drain streams

```
fdt_drain:        drain_org.t - drain_sol.t = dt_drain;
```

Code 4.463: Density of organic drain

```
ifl ref(drain_org.Ash) then
f1rho_drain:      (1.0/drain_org.rho - drain_org.ash_content/
    drain_org.Ash.rho)*(1.0 - feed.ash_content)
                  = (1.0/feed.rho - feed.ash_content/feed.Ash.rho)
                      *(1.0 - drain_org.ash_content);
elsel
f2rho_drain:      1.0/drain_org.rho*(1.0 - feed.ash_content)
                  = (1.0/feed.rho - feed.ash_content/feed.Ash.rho)
                      *(1.0 - drain_org.ash_content);
endifl
```

Code 4.464: Separation efficiencies

```
fsep_eff_inorg: drain_sol.massflow*100.0 = sep_eff_inorg*feed.
    massflow*feed.ash_content;
fsep_eff_org:   drain_sol.massflow*drain_sol.char_content*100.0 =
    sep_eff_org*feed.massflow*(1.0 - feed.ash_content - feed.
    water_content);
```

Code 4.465: Exergy loss

```
fE_loss:          drain_sol.Exergy + drain_org.Exergy + E_loss =
    feed.Exergy;
```

As the definition of the density is compulsory for an organic stream, it is calculated by Code 4.463, for the organic *drain* stream. It has to be mentioned that the equations of Code 4.463 are not made for organic co-separation. The other Code sections are included to calculate further variables of Table 4.54. Figure 4.81 shows the energy and mass balance of this unit.

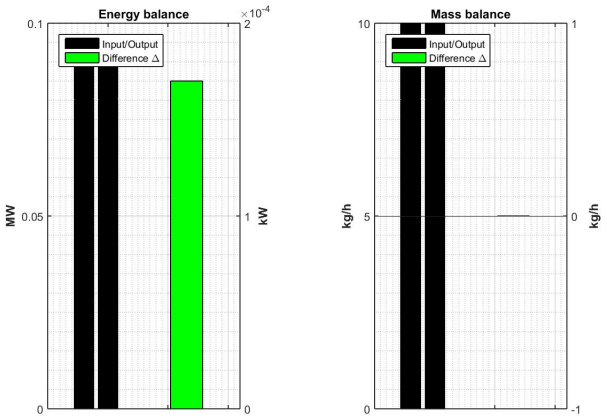


Figure 4.81: Energy and mass balance for the inorganic-organic separator unit

Separator water-organic u_sep_ow_

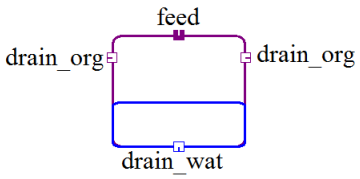


Figure 4.82: Separator water-organic library icon

The model separates water from a water containing organic stream. Ash is optionally possible in the organic stream and remains in organic phase. The organic stream is fed through terminal *feed* and leaves either on the right or on the left side through terminal *drain_org*. The separated water exits through the connector *drain_wat*.

Variables	Description
dp_org	Pressure drop of the organic liquid passing the separator [bar]
dp_wat	Pressure drop between organic feed and water drain [bar]
sep_eff	Separation efficiency describes the water removed in relation to the water fed [%]
E_loss	Exergy loss [kW]

Table 4.55: Separator water-organic model item description

Code 4.466: Energy balance

```
fEnergy: drain_wat.massflow*drain_wat.h_total + drain_org.massflow
    *drain_org.h_total = feed.massflow*feed.h_total;
```

Code 4.467: Mass balances

```
fmass_org: drain_org.massflow*(1.0 - drain_org.water_content -
    drain_org.ash_content) = feed.massflow*(1.0 - feed.
    water_content - feed.ash_content);
ifl ref(feed.Ash) || ref(drain_org.Ash) then
fmass_ash: drain_org.massflow*drain_org.ash_content = feed.
    massflow*feed.ash_content; endifl
fmass_wat: drain_wat.massflow + drain_org.massflow*drain_org.
    water_content = feed.massflow*feed.water_content;
```

To include the energy and mass balance, Code 4.466 and 4.467 are applied. For the model no energy loss is assumed. As already mentioned in the first few lines, the presence of ash in the organic stream is optionally, see Code 4.467.

Code 4.468: Pressure drops

```
fdp_org:        drain_org.p + dp_org = feed.p;
fdp_wat:        drain_wat.p + dp_wat = feed.p;
```

Code 4.469: Temperature equality of drain streams

```
ftemp_eq:        drain_org.t = drain_wat.t;
```

Code 4.470: Density of organic drain

```
frho_drain:      drain_org.rho = 1.0/(drain_org.water_content*
    drain_wat.v + ((1.0 - drain_org.water_content)/(1.0 - feed.
    water_content))*(1.0/feed.rho - feed.water_content*drain_wat.v)
    );
```

Code 4.471: Separation efficiency

```
fsep_eff:        drain_wat.massflow*100.0 = sep_eff*feed.massflow*
    feed.water_content;
```

Code 4.472: Exergy loss

```
fE_loss: drain_wat.Exergy + drain_org.Exergy + E_loss = feed.
    Exergy;
```

Throughout Code 4.469, the equality of both drain stream temperatures is set. As the definition of the density is compulsory for an organic stream, it is calculated by Code 4.470 for the organic drain stream.

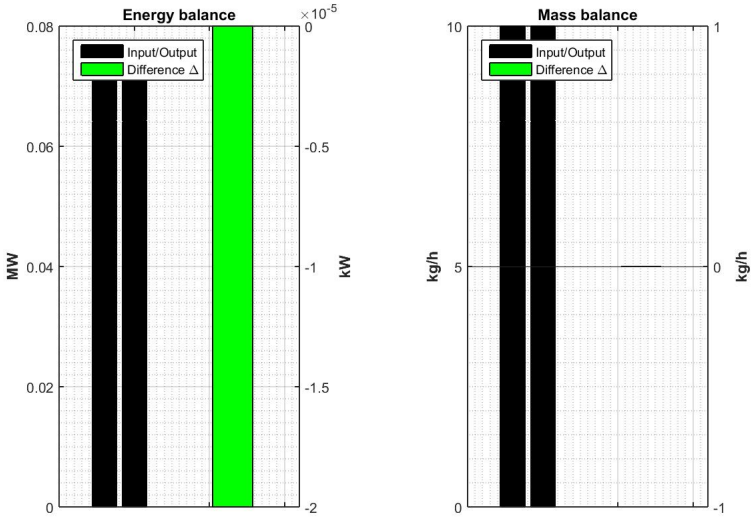


Figure 4.83: Energy and mass balance for the water-organic separator unit

Figure 4.83 shows the correct energy and mass balance for the separation tank below the organic scrubber.

Separator water-emulsion-organic u_sep_ow_3_drain

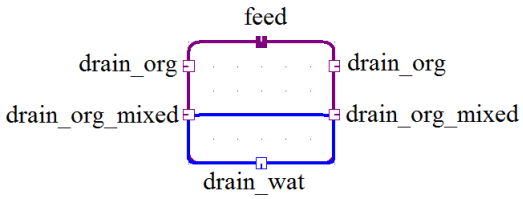


Figure 4.84: Separator water-emulsion-organic library icon

The model separates water from a water containing organic stream, as the unit described right before. Ash is optionally possible in the organic stream and remains in organic phase as well. The organic stream is still fed through terminal *feed* and leaves either on the right or on the left side through terminal *drain_org*. The separated water exits at the connector *drain_wat*. In contrary to the separator before, a third exit has been edited. An emulsion phase -having a low organic content- is led away through terminal *drain_org_mixed*. Additional information on this unit is provided in [60][3.2.3].

Variables	Description
dp_org	Pressure drop of the organic liquid passing the separator [bar]
dp_org_mixed	Pressure drop between organic feed and emulsion phase [bar]
dp_wat	Pressure drop between organic feed and water drain [bar]
rho_org_content	Density of sole organic content(without water and ash) [kg/m^3]
ash_ratio_org	Ash ratio between organic drain and organic feed [%]
ash_ratio_org_mixed	Ash ratio between mixed organic drain and organic feed [%]
E_loss	Exergy loss [kW]

Table 4.56: Separator water-emulsion-organic model item description

Code 4.473: Energy balance

```
fEnergy: drain_wat.massflow*drain_wat.h_total + drain_org_mixed.
    massflow*drain_org_mixed.h_total + drain_org.massflow*drain_org
    .h_total = feed.massflow*feed.h_total;
```

Code 4.474: Mass balance organic stream

```
fmass_org: drain_org.massflow*(1.0 - drain_org.water_content -
    drain_org.ash_content) + drain_org_mixed.massflow*(1.0 -
    drain_org_mixed.water_content - drain_org_mixed.ash_content) =
    feed.massflow*(1.0 - feed.water_content - feed.ash_content);
```

Code 4.475: Mass balance and ratios for the component ash

```
if1 ref(feed.Ash) || ref(drain_org.Ash) || ref(drain_org_mixed.Ash
    ) then
```

```
fmass_ash:        drain_org.massflow*drain_org.ash_content +
    drain_org_mixed.massflow*drain_org_mixed.ash_content = feed.
    massflow*feed.ash_content;
fash_ratio_org:  ash_ratio_org/100 = (drain_org.massflow*drain_org.
    ash_content) / (feed.massflow*feed.ash_content);
fash_ratio_org_mixed: 1 = ash_ratio_org/100 + ash_ratio_org_mixed
    /100;
endifl
```

Code 4.476: Mass balance of water

```
fmass_wat: drain_wat.massflow + drain_org_mixed.massflow*
    drain_org_mixed.water_content + drain_org.massflow*drain_org.
    water_content = feed.massflow*feed.water_content;
```

To grantee the correctness of energy and mass balance, Code 4.473 to 4.476 are applied. As already mentioned in the first few lines, the presence of ash in the organic stream is optionally, see Code 4.475. Dependent on the presence of ash, the calculation of the two ash-ratios of Table 4.56 is done or not. The equations for their calculation are included in Code 4.475.

Code 4.477: Pressure drops

```
fdp_org:          drain_org.p + dp_org = feed.p;
fdp_org_mixed:    drain_org_mixed.p + dp_org_mixed = feed.p;
fdp_wat:          drain_wat.p + dp_wat = feed.p;
```

Code 4.478: Temperature equality of drain streams

```
ftemp_eq:         drain_org.t = drain_wat.t;
ftemp_eq2:        drain_org_mixed.t = drain_wat.t;
```

Code 4.479: Density of organic drain

```
frho_drain:       drain_org.rho = 1.0/(drain_org.water_content*
    drain_wat.v + ((1.0 - drain_org.water_content)/(1.0 - feed.
    water_content))*(1.0/feed.rho - feed.water_content*drain_wat.v)
    );
```

Code 4.480: Separation efficiency

```
ifl ref(feed.Ash) then
frho_org_content1:        rho_org_content = (1-feed.water_content-
    feed.ash_content) / ( (1/feed.rho) - (feed.water_content*
    drain_wat.v) - (feed.ash_content/feed.Ash.rho) );
frho_drain_org1:          drain_org.rho = 1 / ( ((1-drain_org.
    water_content-drain_org.ash_content)/rho_org_content) + (
    drain_org.water_content*drain_wat.v) + (drain_org.ash_content/
    feed.Ash.rho));
frho_drain_org_mixed1:  drain_org_mixed.rho = 1 / ( ((1-
    drain_org_mixed.water_content-drain_org_mixed.ash_content)/
    rho_org_content) + (drain_org_mixed.water_content*drain_wat.v)
    + (drain_org_mixed.ash_content/feed.Ash.rho));
```

```
endifl
ifl !ref(feed.Ash) then
frho_org_content2:      rho_org_content = (1-feed.water_content)
    / ( (1/feed.rho) - (feed.water_content*drain_wat.v) );
frho_drain_org2:        drain_org.rho = 1 / ( ((1-drain_org.
    water_content)/rho_org_content) + (drain_org.water_content*
    drain_wat.v) );
frho_drain_org_mixed2:  drain_org_mixed.rho = 1 / ( ((1-
    drain_org_mixed.water_content)/rho_org_content) + (
    drain_org_mixed.water_content*drain_wat.v) );
endifl
```

Code 4.481: Exergy loss

```
fE_loss:        drain_wat.Exergy + drain_org_mixed.Exergy +
    drain_org.Exergy + E_loss = feed.Exergy;
```

Throughout Code 4.478, the equality of both drain temperatures is set. The density is compulsory for an organic stream, it is calculated by Code 4.479 for the organic drain stream, in dependence if ash is present in the *feed* stream. The energy and mass balance of

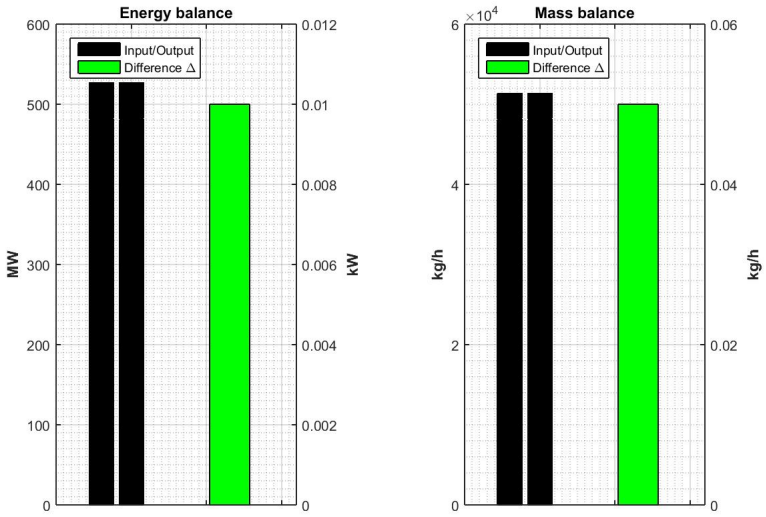


Figure 4.85: Energy and mass balance for the separation drum unit

Figure 4.85 are an indicator for this unit's correct operation.

Separator inorganic-gas u_sep_gs_

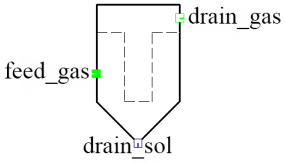


Figure 4.86: Separator inorganic-gas library icon

The model could also be described as a bag filter. It separates inorganic solids (dust) from a gas stream and therefore requires dust referenced in the feed stream (*feed_gas*). If no dust is referenced in the drain gas stream (*drain_gas*), the separation efficiency is automatically set to 100%. Char or tar can be referenced in the gas stream but they are not separated through this unit. The same char and tar global has to be used for the gasous feed and drain stream. No char or adsorbed gases are allowed in the inorganic solid drain stream (*drain_sol*). Additional information on this unit is provided by [76][5.1.4].

Variables	Description
dp_gas	Pressure drop of the gas stream across the filter [bar]
dt_gas_sol	Temperature difference between exiting gas and exiting solid material [°C]
sep_eff	Separation efficiency of the filter in % of entering dust load [%]
q_loss_rel	Relative heat loss in % of sensible heat of the gas incl. dust entering the filter [%]
Q_loss	Heat loss of the filter [kW]
E_loss	Exergy loss [kW]

Table 4.57: Separator inorganic-gas model item description

Code 4.482: Energy balance

```
fh_total:       drain_sol.massflow*drain_sol.h_total + drain_gas.
    massflow*drain_gas.h_total + Q_loss*3600.0 = feed_gas.massflow*
    feed_gas.h_total;
```

Code 4.483: Mass balances

```
fmass_gas:      drain_gas.massflow = feed_gas.massflow;
fmass_dust:     drain_sol.massflow + drain_gas.nvolflow*(drain_gas
    .dust_content/1000) = feed_gas.nvolflow*(feed_gas.dust_content
    /1000);
ifl ref(drain_gas.Char) && ref(drain_gas.Char) == ref(feed_gas.
    Char) then
fmass_char:     drain_gas.char_content = feed_gas.char_content;
endifl
ifl ref(drain_gas.Tar) && ref(drain_gas.Tar) == ref(feed_gas.Tar)
    then
fmass_tar:      drain_gas.tar_content = feed_gas.tar_content;
```

```
endifl
```

The energy and mass balance can be seen in Code 4.482 and 4.483. As described in the paragraph at the beginning of this section, char and tar do not have to be present compulsory in the drain gas stream, see Code 4.483.

Code 4.484: Pressure drop for gas stream

```
fdp_gas:          drain_gas.p + dp_gas = feed_gas.p;
```

Code 4.485: Temperature difference between exiting gas and exiting solid material

```
fdt_gas_sol: drain_sol.t + dt_gas_sol = drain_gas.t;
```

Code 4.486: Specific realtive heat loss of the separator

```
fQ_loss: Q_loss*3600.0 = (feed_gas.massflow*feed_gas.h + feed_gas.
    nvolflow*(feed_gas.dust_content/1000)*feed_gas.h_dust)*
    q_loss_rel/100;
```

The relative heat loss is based on total sensible enthalpy of the gas and dust, entering the separator (Code4.486).

Code 4.487: Split condition for dust according to separation efficiency

```
fsep_eff:         drain_sol.massflow = (sep_eff/100)*feed_gas.
    nvolflow*(feed_gas.dust_content/1000);
```

Code 4.488: Exergy loss

```
fExergy:          drain_sol.Exergy + drain_gas.Exergy + E_loss =
    feed_gas.Exergy;
```

The remaining Code sections from 4.484 to 4.488 are initiated to calculate the remaining variables of Table 4.54.

Figure 4.87 presents the calculated energy and mass balance for this separator unit.

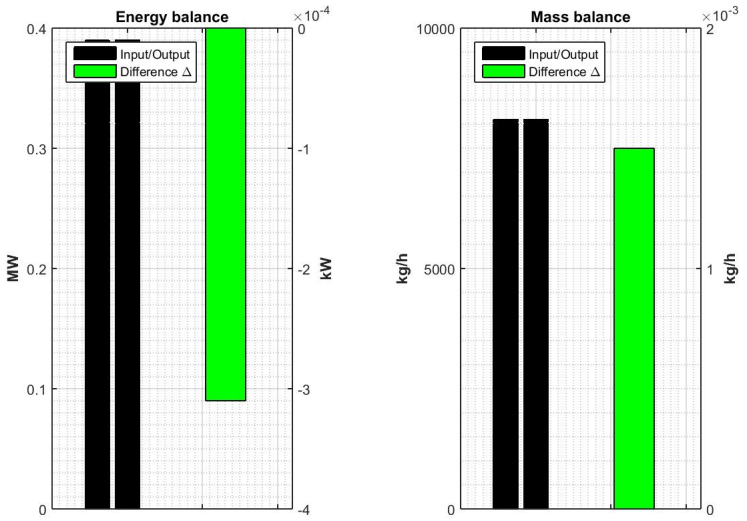


Figure 4.87: Energy and mass balance for the inorganic-gas separation unit

Sink gas u_sink_g_

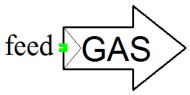


Figure 4.88: Gas sink library icon

The sink models are having more or less a symbolic character. They are available for any global class (ambient, gas, organic, solid) and the connection class water. All of them are similar to each other and therefore the most commonly applied gas sink unit has been taken to described the sink units. All the variables of Table 4.58 representing the values from the feed stream at terminal *feed*. This can be seen by Code 4.489 to 4.493.

Variables	Description
massflow	Mass flow of gas exiting the flow sheet [kg/h]
Mass_total	Total mass flow including dust, char, and tar loads [kg/h]
nvolflow	Standard volume flow of gas exiting the flow sheet [Nm^3/h]
p	Absolute pressure of gas exiting the flow sheet [bar]
t	Temperature of gas exiting the flow sheet [°C]

Table 4.58: Gas sink model item description

Code 4.489: Mass flow gas

```
fmassflow:        massflow = feed.massflow;
```

Code 4.490: Mass_total

```
fMass_total:    Mass_total = feed.Mass_total;
```

Code 4.491: nvolflow

```
fnvolflow:      nvolflow = feed.nvolflow;
```

Code 4.492: Pressure p

```
fp:     p = feed.p;
```

Code 4.493: Temperature t

```
ft:     t = feed.p;
```

Source gas u_source_g_

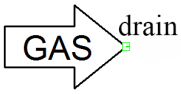


Figure 4.89: Gas source library icon

Like the sink models also the source models are more or less a symbolic objects. They are as well available for any global class (ambient, gas, organic, solid) and the connection class water. Also all of them are similar to each other and the gas model has been taken as it is most commonly applied as well. All variables of Table 4.59 representing the values from the exciting stream terminal *drain*. This can be seen by Code 4.494 to 4.498.

Variables	Description
massflow	Mass flow of gas incoming to the flow sheet [kg/h]
nvolflow	Standard volume flow of gas incoming to the flow sheet [Nm^3/h]
p	Absolute pressure of gas incoming to the flow sheet [bar]
t	Temperature of gas incoming to the flow sheet [°C]

Table 4.59: Gas source model item description

Code 4.494: Mass flow gas

```
fmassflow:      massflow = drain.massflow;
```

Code 4.495: Mass_total

```
fMass_total:    Mass_total = drain.Mass_total;
```

Code 4.496: nvolflow

```
fnvolflow:      nvolflow = drain.nvolflow;
```

Code 4.497: Pressure p

```
fp:     p = drain.p;
```

Code 4.498: Temperature t

```
ft:     t = drain.p;
```

As this unit represents a mass and energy source, neither the mass nor the energy balance are closed.

Splitter gas u_splitter_g_

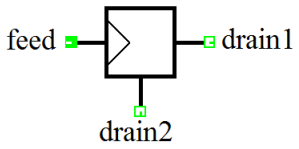


Figure 4.90: Gas splitter library icon

The splitter model is available for the stream classes gas, organic, solid and water. All of them are similar to each other. The gas model has been taken for the description. The model does not contain any item-variables, therefore no explanatory table is listed below.

Code 4.499: Mass balance

```
fmass:  feed.massflow = drain1.massflow+drain2.massflow;
```

Code 4.500: Gas species balances

```
if1     ref(feed.Gas) != ref(drain1.Gas) || ref(feed.Gas) != ref(
    drain2.Gas) || ref(drain1.Gas) != ref(drain2.Gas)
then
f1wAr:  feed.massflow * feed.Gas.wAr = drain1.massflow * drain1.
    Gas.wAr + drain2.massflow * drain2.Gas.wAr;
f1wC2H4:        feed.massflow * feed.Gas.wC2H4 = drain1.massflow *
     drain1.Gas.wC2H4 + drain2.massflow * drain2.Gas.wC2H4;
                        .
                        .
f1wSO2: feed.massflow * feed.Gas.wSO2 = drain1.massflow * drain1.
    Gas.wSO2 + drain2.massflow * drain2.Gas.wSO2;
endif1
```

Code 4.499 ensures the unit's mass balance while Code 4.500 calculates the drain gas composition if just one of the two drain streams is carrying gas. All components except water are considered for the balances. The water content is calculated by the gas global as the sum of all components equals one.

Code 4.501: Dust, char, tar content

```
ifl ref(feed.Dust) then
fdust_cont1:    feed.dust_content = drain1.dust_content;
fdust_cont2:    feed.dust_content = drain2.dust_content;
endifl
ifl ref(feed.Char) then
fchar_cont1:    feed.char_content = drain1.char_content;
fchar_cont2:    feed.char_content = drain2.char_content;
endifl
ifl ref(feed.Tar) then
ftar_cont1:     feed.tar_content = drain1.tar_content;
ftar_cont2:     feed.tar_content = drain2.tar_content;
endifl
```

The dust, char and tar content is calculated via Code 4.501 in dependency if they are present in the *feed* gas stream.

Code 4.502: Pressure equality

```
fpress1:        feed.p = drain1.p;
fpress2:        feed.p = drain2.p;
```

Code 4.503: Temperature equality

```
ftemp1: feed.t = drain1.t;
ftemp2: feed.t = drain2.t;
```

Pressure and temperature equality are set due to Code 4.502 and 4.503.

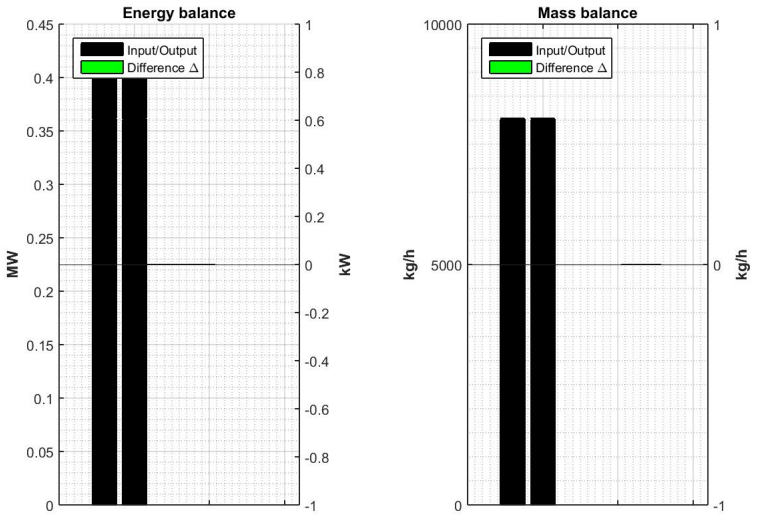


Figure 4.91: Energy and mass balance for the gas splitter model

Figure 4.91 shows the correctness of the splitter's energy and mass balance.

Thermoelectric generator gas-gas u_teg_gg_

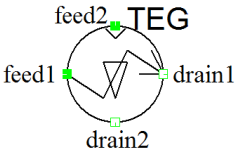


Figure 4.92: Thermoelectric generator library icon

The model describes the conversion of heat into electric energy. The first stream enters at *feed1* and leaves at *drain1*. For the second stream its entrance is called *feed2* and the exit *drain2*.

Switch	Description
Type	Defines heat exchanger type either co- or counter-current
Variables	**Description**
dp_1	Pressure drop of stream 1, which is connected by the zick-zack arrow [bar]
dp_2	Pressure drop of stream 2, which is not connected by the zick-zack arrow in the symbol [bar]
dt_in_1	Temperature difference between the two streams at the entry side of stream 1, which is connected by the zick-zack arrow [°C]
dt_out_1	Temperature difference between the two streams at the exit side of stream 1, which is connected by the zick-zack arrow [°C]
dt_m	Mean temperature difference of heat transfer [°C]
Q_hot	Heat power transferred from hot stream [kW]
Q_cold	Heat power transferred to cold stream [kW]
P_el_DC_spec	Specific electric power output [kW/K]
P_el_DC	Electric power of the thermoelectric generator [kW]
P_el_AC	Alternate current output of the DC/AC inverter [kW]
eta_el_DC	Electric efficiency of thermoelectric generator [%]
eta_inv	Efficiency of DC/AC inverter, heat release of inverter does not enter the energy balance [%]
eta_el_AC	Electric efficiency of the whole TEG-inverter system [%]
E_loss	Exergy loss [kW]

Table 4.60: Thermoelectric generator model item description

Code 4.504: Mass balance

```
fmass1: feed1.massflow = drain1.massflow;
fmass2: feed2.massflow  = drain2.massflow;
```

Code 4.505: Mass balance dust, char and tar

```
if1 ref(feed1.Dust) || ref(drain1.Dust) then
fdust_cont1:    feed1.dust_content = drain1.dust_content;
endif1

if1 ref(feed2.Dust) || ref(drain2.Dust) then
fdust_cont2:    feed2.dust_content = drain2.dust_content;
endif1

if1 ref(feed1.Char) || ref(drain1.Char) then
fchar_cont1:    feed1.char_content = drain1.char_content;
endif1

if1 ref(feed2.Char) || ref(drain2.Char) then
fchar_cont:     feed2.char_content = drain2.char_content;
endif1

if1 ref(feed1.Tar) || ref(drain1.Tar) then
ftar_cont1:     feed1.tar_content = drain1.tar_content;
endif1

if1 ref(feed2.Tar) || ref(drain2.Tar) then
ftar_cont:      feed2.tar_content = drain2.tar_content;
endif1
```

The mass continuity is defined by Code 4.504. The mass continuity of dust, char and tar is defined by Code 4.505 for both streams, in dependency if they are present in the respective stream.

Code 4.506: Pressure drops

```
fdp1:   feed1.p - dp_1 = drain1.p;
fdp2:   feed2.p - dp_2 = drain2.p;
```

Code 4.507: Energy balance

```
fh1:    if (feed1.t < feed2.t) then feed1.massflow*feed1.h_total
   + Q_cold*3600.0 = drain1.massflow*drain1.h_total;
else feed1.massflow*feed1.h_total - Q_hot*3600.0 = drain1.massflow
   *drain1.h_total;
fh2:    if (feed1.t < feed2.t) then feed2.massflow*feed2.h_total
   = drain2.massflow*drain2.h_total + Q_hot*3600.0;
else feed2.massflow*feed2.h_total = drain2.massflow*drain2.h_total
    - Q_cold*3600.0;
```

Code 4.508: Electric power of the thermoelectric generator

```
fP_el_DC:       P_el_DC = Q_hot - Q_cold;
```

Code 4.509: Temperature differences

```
if1 Type == CoCurrent then
fdt_co1: if (feed1.t < feed2.t) then dt_in_1 = feed2.t - feed1.t;
```

```
                                    else dt_in_1 = feed1.t - feed2.t;
fdt_co2: if (feed1.t < feed2.t)          then dt_out_1 = drain2.t -
    drain1.t;                             else dt_out_1 = drain1.t
   - drain2.t;
endif1

if1 Type == CounterCurrent then
fdt_cc1:        if (feed1.t < feed2.t)  then dt_in_1 = drain2.t -
   feed1.t;
else dt_in_1 = drain1.t - feed2.t;
fdt_cc2:        if (feed1.t < feed2.t)  then dt_out_1 = feed2.t -
   drain1.t;
else dt_out_1 = feed1.t - drain2.t;
endif1
```

The pressure drops are calculated by Code 4.506, the energy balance is satisfied by Code 4.507 and the temperature differences of Table 4.60 are defined by Code 4.509 dependent on the selected heat exchanger type.
The heat exchange between two gas streams has been chosen for this documentation.

Code 4.510: Effective temperature difference

```
fdelta_t_eff:   if (abs(dt_in_1/dt_out_1) >= 1.2 || abs(dt_out_1/
   dt_in_1) >= 1.2) && (dt_in_1 > 0) && (dt_out_1 > 0) then
        dt_m = (dt_in_1 - dt_out_1)/ln(dt_in_1/dt_out_1);
else
        dt_m = 0.5*(dt_in_1 + dt_out_1);
```

In Code 4.510 as a first option *dt_m* is calculated as logarithmic average temperature difference the second option is the calculation of the arithmetic average temperature difference.

Code 4.511: Specific electric power output

```
fP_el_DC_spec:  P_el_DC_spec = P_el_DC/dt_m;
```

Code 4.512: Electric efficiency

```
feta_el:        0.01*eta_el_DC*Q_hot = P_el_DC;
```

Code 4.513: Correlation between electric efficiency and mean temperature difference

```
feta_dt:        eta_el_DC*250.0 = 9.0*dt_m;
```

Code 4.514: Alternate current output after DC/AC inversion

```
fP_el_AC:       P_el_AC = 0.01*eta_inv*P_el_DC;
feta_el_AC:     eta_el_AC = 0.01*eta_inv*eta_el_DC;
```

Code 4.515: Exergy loss

```
f_E_loss: P_el_DC + E_loss = feed1.Exergy + feed2.Exergy - drain1.
   Exergy - drain2.Exergy;
```

From Code 4.511 to 4.515 the remaining variables of Table 4.60 are defined.
Figure 4.93 presents the results of an energy and mass balance carried out.

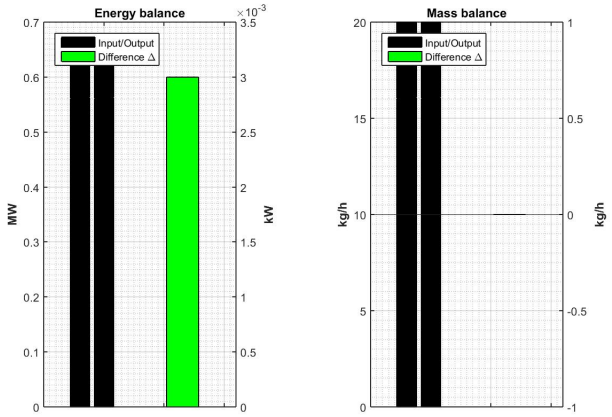


Figure 4.93: Energy and mass balance for the thermoelectric generator model

Transformer steam-gas u_trans_wg_

Figure 4.94: Transformer steam-gas library icon

The model describes the conversion of a water stream (*feed*) into a gas stream *drain*. In the water stream, the water is already in gaseous phase to obtain a correct visualization, by this unit the steam stream is presented as a gas stream.

Variables	Description
Q_calc	Heat difference between ideal gas and real water calculation [kW]

Table 4.61: Transformer steam-gas model item description

Code 4.516: Mass balance

```
fmass:   feed.massflow = drain.massflow;
```

Code 4.517: Pressure drop

```
fp:      feed.p = drain.p;
```

Code 4.518: Energy balance

```
fh_total: feed.massflow*feed.h_total = drain.massflow*drain.
    h_total;
fq_calc: Q_calc = 0;
```

Code 4.519: Setting values in gas global of drain stream

```
ifl ref(drain.Gas) then
fAr:    drain.Gas.yAr = 0.0;
fC2H4:  drain.Gas.yC2H4 = 0.0;
fC2H6:  drain.Gas.yC2H6 = 0.0;
fC3H8:  drain.Gas.yC3H8 = 0.0;
fCH4:   drain.Gas.yCH4 = 0.0;
fCO:    drain.Gas.yCO = 0.0;
fCO2:   drain.Gas.yCO2 = 0.0;
fH2:    drain.Gas.yH2 = 0.0;

fH2S:   drain.Gas.yH2S = 0.0;
fHCl:   drain.Gas.yHCl = 0.0;
fHCN:   drain.Gas.yHCN = 0.0;
fN2:    drain.Gas.yN2 = 0.0;
fN2O:   drain.Gas.yN2O = 0.0;
fNH3:   drain.Gas.yNH3 = 0.0;
fNO:    drain.Gas.yNO = 0.0;
fO2:    drain.Gas.yO2 = 0.0;
fSO2:   drain.Gas.ySO2 = 0.0;
endifl
```

The unit's mass balance is fixed by Code 4.516, its energy balance by Code 4.518 where *Q_calc* is assumed as zero. The feed and drain pressure are set equal by Code 4.517. The drain stream's gas composition is defined by Code 4.519 where the water content is not listed as it is calculated by the gas global (Eq. 4.45), the sum of all molar fractions is 100%. Figure 4.95 shows an energy and mass balance related to this unit.

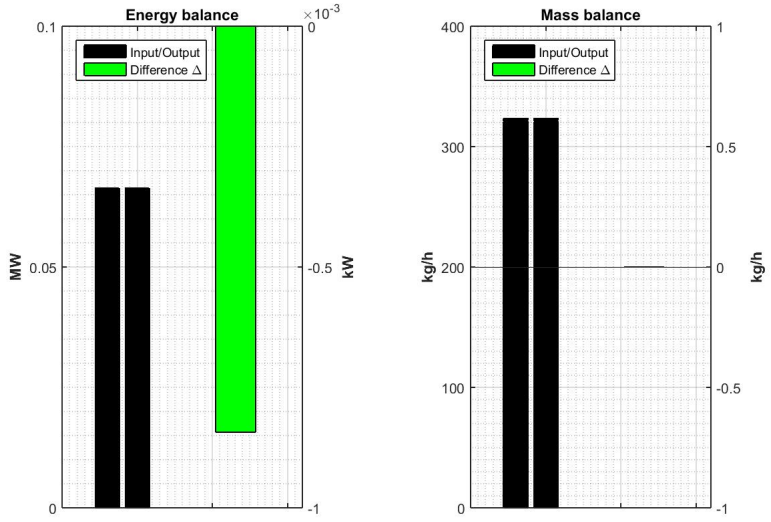


Figure 4.95: Energy and mass balance for the steam-gas transformer model

Trap u_trap_w_

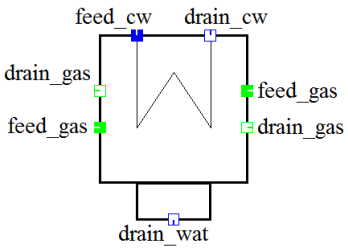


Figure 4.96: Cooling trap library icon

The cooling trap is cooling a gas stream (feed_gas/drain_gas) by partly evaporating water and heating up the cooling water (feed_cw/drain_cw). Steam in the gas stream is condensing and removed through exit *drain_wat*.

Switch	Description
Type	Heat exchanger type either co- or counter-current
Variables	**Description**
Q_trans	Heat transfer to the cooling liquid [kW]
dt_in_cw	Temperature difference of the entering cooling water [°C]
dt_out_cw	Temperature difference of the exiting water [°C]
dp_gas	Pressure drop for gas stream [bar]
dp_cw	Pressure drop for water stream [bar]
dp_drain	Pressure difference between the condensed water and gas stream exiting the trap [bar]
dt_drain	Temperature difference between exiting gas and exiting water [°C]
p_sat	Saturation pressure of the exiting gas [bar]
phi_rel	Relative humidity of the exiting gas stream [%]
kA	Heat transfer coefficient multiplied by transfer area [kW/K]
E_loss	Exergy loss in the quench [kW]

Table 4.62: Cooling trap model item description

Code 4.520: Energy balance and transported heat

```
fEnergy: drain_gas.massflow*drain_gas.h_total + drain_wat.massflow
    *drain_wat.h_total + drain_cw.massflow*drain_cw.h_total =
    feed_gas.massflow*feed_gas.h_total + feed_cw.massflow*feed_cw.
    h_total;
fQ_trans: Q_trans * 3600.0 = drain_cw.massflow * drain_cw.h_total
    - feed_cw.massflow * feed_cw.h_total;
```

Code 4.521: Mass balance dust, char and tar

```
if1 ref(drain_gas.Dust) && ref(feed_gas.Dust) then
fmass_dust:      drain_gas.nvolflow*(drain_gas.dust_content/1000.0)
    = feed_gas.nvolflow*(feed_gas.dust_content/1000.0);
```

```
endifl

ifl ref(drain_gas.Char) && ref(feed_gas.Char) then
fmass_char:      drain_gas.nvolflow*(drain_gas.char_content/1000.0)
    = feed_gas.nvolflow*(feed_gas.char_content/1000.0);
endifl

ifl ref(drain_gas.Tar) && ref(feed_gas.Tar) then
fmass_tar:       drain_gas.nvolflow*(drain_gas.tar_content/1000.0)
   = feed_gas.nvolflow*(feed_gas.tar_content/1000.0);
endifl
```

Code 4.522: Mass balance cooling water

```
f_masscw:                  feed_cw.massflow = drain_cw.massflow;
```

Code 4.523: Species balances

```
ifl ref(drain.Gas) then
fmass_H2O: drain_gas.massflow*drain_gas.Gas.wH2O + drain_wat.
    massflow = feed_gas.massflow*feed_gas.Gas.wH2O;

fmass_Ar: drain_gas.massflow*drain_gas.Gas.wAr = feed_gas.massflow
    *feed_gas.Gas.wAr;
fmass_C2H4: drain_gas.massflow*drain_gas.Gas.wC2H4 = feed_gas.
    massflow*feed_gas.Gas.wC2H4;
fmass_C2H6: drain_gas.massflow*drain_gas.Gas.wC2H6 = feed_gas.
    massflow*feed_gas.Gas.wC2H6;
fmass_C3H8: drain_gas.massflow*drain_gas.Gas.wC3H8 = feed_gas.
    massflow*feed_gas.Gas.wC3H8;
fmass_CH4: drain_gas.massflow*drain_gas.Gas.wCH4 = feed_gas.
    massflow*feed_gas.Gas.wCH4;
fmass_CO: drain_gas.massflow*drain_gas.Gas.wCO = feed_gas.massflow
    *feed_gas.Gas.wCO;
fmass_CO2: drain_gas.massflow*drain_gas.Gas.wCO2 = feed_gas.
    massflow*feed_gas.Gas.wCO2;
fmass_H2: drain_gas.massflow*drain_gas.Gas.wH2 = feed_gas.massflow
    *feed_gas.Gas.wH2;
fmass_H2S: drain_gas.massflow*drain_gas.Gas.wH2S = feed_gas.
    massflow*feed_gas.Gas.wH2S;
fmass_HCl: drain_gas.massflow*drain_gas.Gas.wHCl = feed_gas.
    massflow*feed_gas.Gas.wHCl;
fmass_HCN: drain_gas.massflow*drain_gas.Gas.wHCN = feed_gas.
    massflow*feed_gas.Gas.wHCN;
fmass_N2: drain_gas.massflow*drain_gas.Gas.wN2 = feed_gas.massflow
    *feed_gas.Gas.wN2;
fmass_N2O: drain_gas.massflow*drain_gas.Gas.wN2O = feed_gas.
    massflow*feed_gas.Gas.wN2O;
fmass_NH3: drain_gas.massflow*drain_gas.Gas.wNH3 = feed_gas.
    massflow*feed_gas.Gas.wNH3;
fmass_NO: drain_gas.massflow*drain_gas.Gas.wNO = feed_gas.massflow
    *feed_gas.Gas.wNO;
fmass_O2: drain_gas.massflow*drain_gas.Gas.wO2 = feed_gas.massflow
    *feed_gas.Gas.wO2;
```

```
fmass_SO2: drain_gas.massflow*drain_gas.Gas.wSO2 = feed_gas.
    massflow*feed_gas.Gas.wSO2;
endifl
```

Code 4.524: Pressure drops

```
fdp_gas:        drain_gas.p + dp_gas = feed_gas.p;
fdp_wat:        drain_cw.p + dp_cw = feed_cw.p;
```

Code 4.525: Pressure difference between exiting gas and exiting water

```
fdp_drain:      drain_gas.p + dp_drain = drain_wat.p;
```

Code 4.526: Temperature difference between exiting gas and exiting water

```
fdt_drain:      drain_gas.t = drain_wat.t + dt_drain;
```

Code 4.527: Exergy loss

```
fE_loss:        drain_gas.Exergy + drain_wat.Exergy + drain_cw.
    Exergy + E_loss = feed_gas.Exergy + feed_cw.Exergy;
```

Code 4.528: Condensate stream

```
f_codensate: if (drain_wat.massflow >0.0) then
        drain_gas.Gas.yH2O*drain_gas.p = feed_cw.wfp0t(drain_gas.t
            );
else
        drain_wat.massflow = 0.0;
```

If the mass flow of water (*drain_wat*) is existing, 100% saturation is assumed, if not its mass flow is set to zero, see Code 4.528.

Code 4.529: Relative humidity of the gas stream

```
f_phi_rel:              phi_rel = drain_gas.p / p_sat * 100.0;
```

Code 4.530: Saturation pressure of drain gas stream

```
fp_sat: p_sat = drain_gas.wfp0t(drain_gas.t)/drain_gas.Gas.yH2O;
```

Code 4.531: Temperature differences dependent on heat exchanger type

```
ifl Type == CoCurrent then
fdt_co1: if (feed_cw.t < feed_gas.t)    then dt_in_cw = feed_gas.t
    - feed_cw.t;
else dt_in_cw = feed_cw.t - feed_gas.t;
fdt_co2: if (feed_cw.t < feed_gas.t)    then dt_out_cw = drain_gas
    .t - drain_cw.t;
else dt_out_cw = drain_cw.t - drain_gas.t;
endifl

ifl Type == CounterCurrent then
```

```
fdt_cc1:        if (feed_cw.t < feed_gas.t) then
        dt_in_cw = drain_gas.t - feed_cw.t;
else dt_in_cw = drain_cw.t - feed_gas.t;
        fdt_cc2:        if (feed_cw.t < feed_gas.t) then
        dt_out_cw = feed_gas.t - drain_cw.t;
else dt_out_cw = feed_cw.t - drain_gas.t;
endif
```

Code 4.532: Effective temperature difference

```
fdelta_t_eff:   if (abs(dt_in_cw/dt_out_cw) >= 1.2 || abs(
    dt_out_cw/dt_in_cw) >= 1.2) && (dt_in_cw > 0) && (dt_out_cw >
    0) then
        Q_trans*ln(dt_in_cw/dt_out_cw)/(dt_in_cw - dt_out_cw) = kA
            ;
else
        Q_trans = kA*0.5*(dt_in_cw + dt_out_cw);
```

Remaining variables of Table 4.62 are calculated via Code 4.529 to 4.532.

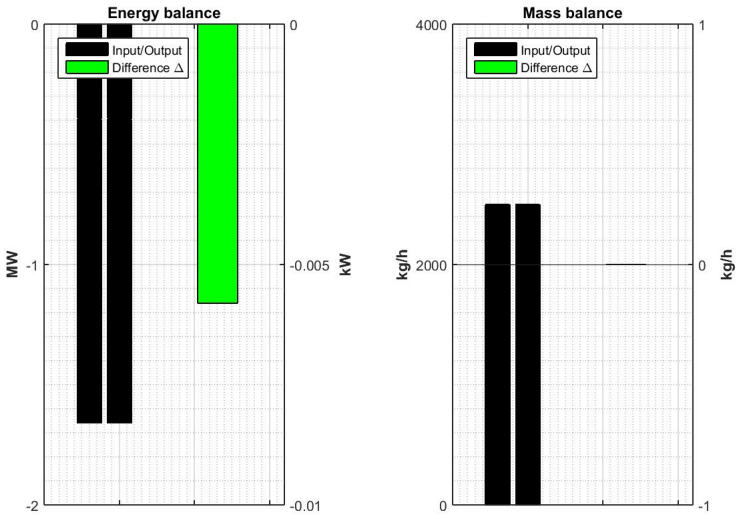


Figure 4.97: Energy and mass balance for the cooling trap unit

Figure 4.97 shows this unit's energy and mass balance.

Turbine gas u_turb_g_

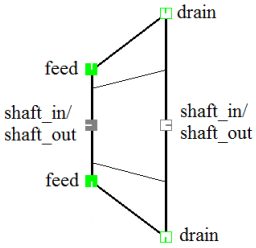


Figure 4.98: Gas turbine library icon

Model gas_turbine_std expands a gas stream. Dust, char, and tar can be referenced optionally. If referenced, the same global objects have to be used in the *feed* and *drain* stream. The shaft in- and output is called *shaft_in/shaft_out*. Again, as always feed and drain terminal for a shaft connection are having the same name due to rotation mode. Even when rotating the icon in PSE, shaft_in is always connected from the left side. The various mass and energy balances as well as the variables of Table 4.63 are defined in the different code sections below the table. There is also a water turbine available. Its code is not listed separately as it is similar to the one of this gas turbine. Additional information on the gas turbine model can be found in [76][5.1.2].

Variables	Description
dp	Pressure difference between feed and drain [bar]
press_ratio	Pressure ratio p_drain/p_feed [-]
eta_s	Isentropic efficiency [%]
eta_m	Mechanical efficiency [%]
P_rotor_disc	Power at the rotor disc [kW]
ts	Hypothetical temperature after isentropic expansion [°C]
E_loss	Exergy loss [kW]

Table 4.63: Gas turbine model item description

Code 4.533: Pressure drop

```
fdp:    drain.p = feed.p + dp;
```

Code 4.534: Mass balance

```
fmass:  feed.massflow = drain.massflow;
```

Code 4.535: Mass balances dust, char, tar

```
ifl ref(feed.Dust) || ref(drain.Dust) then fmass_dust:  feed.
    dust_content = drain.dust_content; endifl
ifl ref(feed.Char) || ref(drain.Char) then fmass_char:  feed.
    char_content = drain.char_content; endifl
```

```
ifl ref(feed.Tar) || ref(drain.Tar) then fmass_tar:      feed.
    tar_content = drain.tar_content;    endifl
```

Code 4.536: Hypothetical temperature after isentropic expansion implicitely determined by isentropic enthalpy difference

```
fts:     feed.h - (feed.h - drain.h)/(eta_s/100) = drain.Gas.gfht(
    ts);
```

Code 4.537: Irreversible expansion

```
fturb:   feed.s = drain.Gas.gfspt(drain.p, ts);
```

Code 4.538: Energy balance

```
ifl ref(shaft_in) && ref(shaft_out) then
f1h:     (feed.h_total - drain.h_total)* feed.massflow*eta_m/100 =
    (shaft_out.power - shaft_in.power)*3600.0;
f1_E_loss:       drain.Exergy + shaft_out.power + E_loss = feed.
    Exergy + shaft_in.power;
endifl

ifl ref(shaft_in) && !ref(shaft_out) then
f2h:     (feed.h_total - drain.h_total)* feed.massflow*eta_m/100 =
    - shaft_in.power*3600.0;
f2_E_loss:       drain.Exergy + E_loss = feed.Exergy + shaft_in.
    power;
endifl

ifl !ref(shaft_in) && ref(shaft_out) then
f3h:     (feed.h_total - drain.h_total)* feed.massflow*eta_m/100 =
    shaft_out.power*3600.0;
f3_E_loss:       drain.Exergy + shaft_out.power + E_loss = feed.
    Exergy;
endifl
```

The energy balance (Code 4.538) is calculated in dependency if a shaft is connected from just one side or both.

Code 4.539: Power of rotor disc

```
f_p_rotor_disc: P_rotor_disc = (feed.h_total - drain.h_total)*feed
    .massflow/3600;
```

Code 4.540: Pressure ratio

```
fpress_ratio:    drain.p*press_ratio = feed.p;
```

Figure 4.99 the correct operation of the turbine model through a correct energy and mass balance.

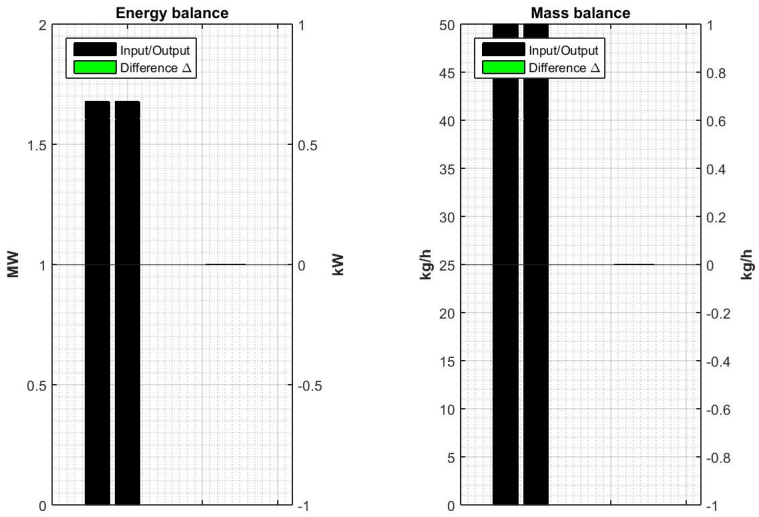


Figure 4.99: Energy and mass balance for the turbine model

Valve gas u_valve_g

Figure 4.100: Valve library icon

The valve model is kept simple. It contains a gas inlet called *feed* and an outlet named *drain*. Its mass and energy balances as well as the variables of Table 4.64 are defined in the different code sections below the table. There is also a water valve available. The code is not listed separately as it is similar to the one of this gas valve.

Variables	Description
dp	Pressure difference between feed and drain [bar]
dt	Temperature drop in valve [°C]
q_loss_rel	Loss of heat relative to sensible gas heat in feed stream [%]
Q_loss	Heat loss [kW]
E_loss	Exergy loss [kW]

Table 4.64: Valve model item description

Code 4.541: Energy balance

```
fh_total:          drain.massflow*drain.h_total + Q_loss*3600.0 =
    feed.massflow*feed.h_total;
```

Code 4.542: Mass balance

```
f_mass: drain.massflow = feed.massflow;
```

Code 4.543: Mass balances dust, char, tar

```
ifl ref(feed.Dust) && ref(drain.Dust) then
f_dust_content: drain.dust_content = feed.dust_content;
endifl
ifl ref(feed.Char) && ref(drain.Char) then
f_char_content: drain.char_content = feed.char_content;
endifl
ifl ref(feed.Tar) && ref(drain.Tar) then
f_tar_content:  drain.tar_content = feed.tar_content;
endifl
```

Code 4.544: Pressure drop

```
fdp:    feed.p = drain.p + dp;
```

Code 4.545: Temperature drop

```
fdt:    feed.t = drain.t + dt;
```

Code 4.546: Heat loss

```
fq_loss:        Q_loss*3600.0 = feed.massflow*feed.h*q_loss_rel
    /100.0;
```

Code 4.547: Exergy loss

```
fExergy:        drain.Exergy + E_loss = feed.Exergy;
```

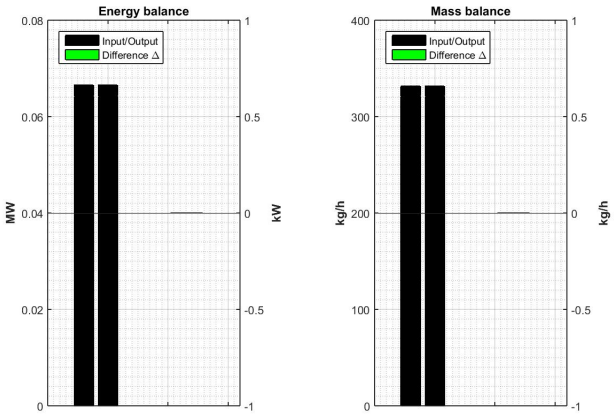


Figure 4.101: Energy and mass balance for the valve model

Figure 4.101 provides an illustration of a calculated energy and mass balance.

5 Conclusion and Outlook

The huge potential of wind- and sun-energy for electricity production requires energy storage strategies due to their fluctuating availability. One of the most promising solutions is the Power-to-Gas (PtG) approach. Excess electricity is stored as chemical energy. The potential for SNG storage in the natural gas grid is of about 400 TWh just for the German grid.[11] The Austrian Rohöl-Aufsuchungsgesellschaft declared its natural gas storage potential with 63.9 TWh, in 2014.[80]

For the PtG concept of Section 3.1, the required carbon for methanation is provided via DFB gasification at a cold gas efficiency of 75%$_{db}$. For the combustion reactor -of the dual fluidized bed gasifier- the OxyFuel process is implemented. The oxygen is delivered through electrolysis. The produced amount of oxygen exceeds the required amount -for combustion- by a factor of three. For exclusive SNG production, the most detailed overall efficiency is 58-60% and 75-77% if waste heat is recovered for district heating. The SNG output is 1690 Nm^3/h with a methane content of more than 98% and a Wobbe-index of 14.62 kWh/m^3. The generated SNG therefore matches requirements for natural gas grid injection.

The simulation results for the Power-to-Gas concept of CO_2 supply through biogas are a plausible forecast for the planned process. The actual results of the test facility, of methanation and SNG purification by a membrane, are not published yet. They are provided as personal information and therefore they cannot be presented in this work. The overall efficiency for SNG as the only product is 55% for a SNG-methane content of above 98% and a Wobbe-index of 14.56 kWh/m^3. The main purpose of the membrane -to lower the hydrogen and carbon dioxide content for grid injection- is perfectly met. The calculated sensitivities of 5.3 for carbon dioxide and 9.6 for hydrogen already indicate an appreciable methane permeation through the membrane. As the permeate stream is recovered, the high permeability of methane is not an exclusion criteria for the membrane unit as final gas cleaning step.
The heating requirements are met by the process' waste heat, on a proximate calculation. However, a detailed heat recovery investigation was not conducted.

The comparison of simulation results and literature data (Section 3.3) shows slightly better overall efficiencies for carbon supply through biomass gasification than digestion. Nevertheless, gasification of renewable solid feedstock is a key technology for the generation of biobased fuels, chemical and materials which makes the power plant more flexible and profitable if no excess hydrogen is available (Figure 1.4).[55] This is in contrast to fermentation of biomass where just methane is available for further application and the co-generated CO_2 has to be stored or released to the environment.

This work's simulation results have shown the perfect fit of an DFB gasifier in a PtG system. High efficiencies are calculated for any system boundary constraints. Also the flexibility in times of no excess electricity has been highlighted. Data for all the applied process unit's is available through experimental results at TU Wien. Therefore, this dissertation should be a first step for further investigations on the combination of carbon delivery by DFB gasification for Power to Gas concepts.

For currently available electrolyser sizes, it is sufficient to store excess hydrogen and oxygen in gas tanks. If large gas storage facilities are nearby, especially hydrogen could be injected into e.g. salt caverns but so far it is not mandatory to use them or to liquefy the electrolysis products.

The basic elements of the Power to Gas concept already operate at a high level of efficiency and reliability. No significant increase in efficiency for AEL and PEM electrolysis technology as well as the methanation process is expected.
If HTEL technology will reach commercial status, it would be very interesting to include it into one of these strongly exothermic PtG plants. Further challenging tasks for the future will be upscaling and to proof operation reliability of PtG plants.
For the DFB gasification research is in progress on alternative, renewable feedstocks and to further increase the hours of operation.

So far there are still no regulations on the CO content for SNG injection into the natural gas grid, in the ÖVGW G31 directive but neither for other countries. Very low CO limits could result in the installation of pressure swing adsorption units as final SNG cleaning step or to include a downstream CO-shift rector. The ability of CO fine-removal through a membrane could also be considered as well.
The limitation of SNG's hydrogen content is different in each European country. In order to connect the separate natural gas grids, one common European directive has to be established.

For almost all the BG_Lib process units it has been proven that the model units are meeting the mass- and energy-balance. Just the propane reforming reaction of the u_chemeq_ref_ model shows an error for the calculation of its Gibbs free enthalpy. This error does not create any conflict for this dissertation but has to be eliminated within the simprop.dll-file for future work. For this work the methanation reactor model is essential. Its validation has shown agreement for chemical equilibrium with the results of the software tool HSC Chemistry 6.0.
However, the documentation should provide a good basis for future work with the library file.

Bibliography

[1] Solarer Wasserstoff in Mecklenburg-Vorpommern Utopie oder Zukunftstechnologie.

[2] Pinch Analysis: For the Efficient Use, 2003.

[3] Energiepark Mainz: Erstes Elektrolysesystem eingetroffen. Web, Mai 2015.

[4] June 2016.

[5] K. Agbli, M. Pera, D. Hissel, O. Rallieres, C. Turpin, and I. Doumbia. Multiphysics simulation of a PEM electrolyser: Energetic Macroscopic Representation approach. *International Journal of hydrogen energy*, 36(36):1382–1398, 2011.

[6] Albrecht. Wärmeübertragung mit Salzschmelzen - Für hohe Temperaturen. *cav chemie anlagen verfahren*, 10, 2000.

[7] Alexander Burcat. Third Millennium Ideal Gas and Condensed Phase Thermochemical Database for Combustion. Online, July 2001.

[8] G. A. Almansa, L. Rabou, C. van der Meijden, and A. van der Drift. ECN System for MEthanation (ESME). In *23rd European Biomass Conference & Exhibition, Vienna*, 2015.

[9] H. D. Baehr and S. Kabelac. *Thermodynamik - Grundlagen und technische Anwendungen*. Springer, 2000.

[10] S. Bajohr and M. Götz. Dreiphasen-Methanisierung als innovatives Element der PtG-Prozesskette. *gwf-Gas/Erdgas*, 05:328–335, 2012.

[11] S. Bajohr, M. Götz, F. Graf, and F. Ortloff. Speicherung von regenerativ erzeugter elektrischer Energie in der Erdgasinfrastruktur. *GWF Gas Erdgas*, 04:200–210, 2011.

[12] Balticare. Delivering the benefits of cooling towers and evaporative condensers. Online, April 2016.

[13] Bär K. and Mörs F. and Götz M. and Graf F. Vergleich der biologischen und katalytischen Methanisierung für den Einsatz bei PtG-Konzepten. *gwf - Gas*, 7:466–473, 2015.

[14] Bartholomew, Agrawal, and Katzer. Sulfur Poisoning of Metals. *Advances in Catalysis*, 31:135–242, 1981.

[15] P. Baudet, P. Castelain, O. Baudoin, and P. SA. Sequential modular approach or global approach? And why not both? Process Simulation and Optimization.

[16] Benedikt, Kraussler, Konlechner, Bosch, Hackel, and Hofbauer. Polygeneration at the biomass steam gasifcation plant Oberwart. In *23rd European Biomass Conference and Exhibitionition*, 2015.

[17] R. Berger and K. Hein. Session II Gaserzeugung aus Biomasse / Gasreinigung, 2003.

[18] P. Biegger and M. L. Aaron Felde and. Entwicklung eines katalytischen Prozesses zur Methanisierung von CO_2 aus industriellen Quellen.

[19] Boerrigter and den Ui and Calis. Green diesel from biomass by Fischer-Tropsch synthesis: New insights in gas cleclean and process design. In *PGBW Expert Meeting*, 2003.

[20] K. Bosch. *Scale Up der Dampf - Wirbelschicht - Biomassevergasung : vom Technikumsmaßstab in den industriellen Maßstab*. PhD thesis, TU Wien, 2007.

[21] U. Bossel, B. Eliasson, and G. Taylor. The Future of the Hydrogen Economy: Bright or Bleak. In *Fuel Cell Seminar*, 2003.

[22] T. Brauer. Kompaktes 1 MW - PEM - Wasserelektrolysesystem Regenerativer Wasserstoff für Mobilität und Energiespeicherung. Presentation, June 2013.

[23] J. Brellochs. *Experimentelle Untersuchung und Prozess-Simulation der AER-Biomassevergasung zur Erzeugung eines regregenerativen Erdgassubstitutes*. PhD thesis, Universität Stuttgart, 2014.

[24] A. Brisse, J. Schefold, and M. Zahid. High temperature water electrolysis in solid oxide cells. *International journal of hydrogen energy*, 33:5375–5382, 2008.

[25] R. C. Brown. *Biorenewable Resources - Engineering New Products from Agriculture*. Iowa State Press, 2003.

[26] C. Buck. Wasserstoff macht Karriere. *Pictures of the Future*, Frühjahr:100–102, 2012.

[27] A. Burcat and B. Ruscic. Third Millennium Ideal Gas and Condensed Phase Thermochemical Database for Combustion with Updates from Active Thermochemical Tables. Technical report, The University of Chicago for the U. S. Department of Energy, 2005.

[28] D. Carbon. Aktivkohle und ihre anwebdung.

[29] C.M. van der Meijden and B.J. Vreugdenhil and L.P.L.M. Rabou and G. Aranda Almansa. PRODUCTION OF BIO-METHANE FROM WOODY BIOMASS USING THE MILENA GASIFICATION TECHNOLOGY. Technical report, ECN, 2015.

[30] D. Bacovsky and M. Dallos and M. Wörgetter. Status of 2nd Generation Biofuels Demonstration Facilities in June 2010. In *A REPORT TO IEA BIOENERGY TASK 39*, 2010.

[31] David Konlechner. *Aufbereitung von Wasserstoff aus der Biomassevergasung mittels Membrantechnologie*. PhD thesis, Technische Universität Wien, 2015.

[32] V. der Meijen. *Development of the MILENA gasification technology for the production of Bio-SNG*. PhD thesis, Technische Universiteit Eindhofen, 2010.

[33] Donau Carbon. Aktivkohle zur Abscheidung von Schwefelwasserstoff.

[34] F. N. R. e. V. Biogas basisdaten deutschland, 2008.

[35] H. Eichlseder and M. Klell. *Wasserstoff in der Fahrzeugtechnik*. Springer Vieweg, 2012.

[36] B. Epple, R. Leithner, W. Linzer, and H. Walter. *Simulation von Kraftwerken und wärmetechnischen Anlagen*. Springer-Verlag, 2009.

[37] J. Fluch, W. Glatzl, and C. Brunner. HANDBUCH ZUM klimaaktiv PINCH-TOOL, 2014.

[38] Förderprogramm Forschung und Entwicklung zur Optimierung der energetischen Biomassenutzung. FuE-Plattform "Biomass-to-Gas" - Energetische Nutzung biogener Reststoffe mit AER-Technologie zur Poly-Generation von Strom, Wasserstoff, Erdgassubstitut und Wärme. Technical report, ZSW, 2013.

[39] A. Friedl, W. Wukovits, and M. Harasek. Teching notes of the university lecture Prozesssimulation, 2012.

[40] H. Friedmann. Der einfluss der temperatur bei der biogasproduktion. online, September 2016.

[41] M. Fuchs, R. Rauch, and H. Hofbauer. Report on fundamental studies on H2S and HCl removal. Technical report, Vienna University of Technology, 2007.

[42] G.L. Macpherson. CO2 distribution in groundwater and the impact of groundwater extraction on the global C cycle. *Chemical Geology*, 26:328–336, 2009.

[43] M. Götz. *Methanisierung im Dreiphasen-Reaktor*. PhD thesis, Karlsruher Institut für Technologie, 2015.

[44] F. Graf, A. Krajete, and U. Schmack. Techno-ökonomische Studie zur biologischen Methanisierung bei Power-to-Gas-Konzepten. Technical report, DVGW, 2014.

[45] S. Graf. Synergetische Nutzung von fluktuierender Windenergie und gespeicherter Energie in Form von Biomasse. Master's thesis, TU Wien, 2015.

[46] S. Hemetsberger. Literaturrecherche zur Trennung von H2/CO2 Mischungen. Master's thesis, Vienna University of Technology, 2010.

[47] S. Heyne, M. C. Seemann, and S. Harvey. Integration study for alternative methanation technologies for the production of synthetic natural gas from gasified biomass. *CHEMICAL ENGINEERING TRANSACTIONS*, 21:409–410, 2010.

[48] S. Heyne, H. Thunman, and S. Harvey. Integration aspects for synthetic natural gas production from biomass based on a novel indirect gasification concept. In *11th Conference on Process Integration, Modelling and Optimisation for Energy Saving and Pollution Reduction PRES, 24 - 28 August 2008, Prague, Czech Republic*, 2008.

[49] S. Heyne, H. Thunman, and S. Harvey. Extending existing CHP plants for SNG production a process integration study. *International Journal of Energy Research*, 36:670–681, 2012.

[50] H. Hofbauer, R. Reinhard, B. Klaus, K. Reinhard, and A. Christian. Biomass CHP Plant Güssing: A Success Story.

[51] H.T. BI and J. R. GRACE. FLOW REGIME DIAGRAMS FOR GAS-SOLID FLUIDIZATION AND UPWARD TRANSPORT. *International Journal of Multiphase Flow*, 21:1229–1236, 1995.

[52] J. Hüttenrauch and G. Müller-Syring. Zumischung von Wasserstoff zum Erdgas. *Energie - Wasser Praxis*, 10:68–71, 2010.

[53] J. O. Jensen, Q. Li, and N. J. Bjerrum. The energy efficiency of onboard hydrogen storage. Technical report, Technical University of Denmark, 2010.

[54] C. Jünger, A. Kreuzeder, G. Soukup, C. Pfeifer, and H. Hofbauer. DRYING OF BIOMASS - INFLUENCE OF FUEL WATER CONTENT ON THE DUAL FLUIDISED BED STEAM GASIFICATION PROCESS. Success and Visions for Bioenergy. TU Wien.

[55] Jürgen Karl. *Dezentrale Energiesysteme: Neue Technologien im liberalisierten Energiemarkt*. Oldenbourg Wissenschaftsverlag, 2006.

[56] M. Kaltschmitt, H. Hartmann, and H. Hofbauer. *Energie aus Biomasse: Grundlagen, Techniken und Verfahren*. Springer Berlin Heidelberg, 2009.

[57] B. Kamm, P. R. Gruber, and M. Kamm. *Biorefineries - Industrial Processes and Products*. Wiley - VCH, 2006.

[58] J. Kopyscinski. *Production of synthetic natural gas in a fluidized bed reactor*. PhD thesis, Paul Scherrer Institut, 2010.

[59] J. Kopyscinski, T. J. Schildhauer, and S. M. Biollaz. Production of synthetic natural gas (SNG) from coal and dry biomass - A technology review from 1950 to 2009. *Fuel*, 89:1763–1783, 2010.

[60] J. Kotik. *Über den Einsatz von Kraft - Wärme - Kopplungsanlagen auf Basis der Wirbelschicht - Dampfvergasung fester Biomasse am Beispiel des Biomassekraftwerks Oberwart*. PhD thesis, Vienna University of Technology, 2010.

[61] Landesanstalt für Umweltschutz Baden-Württemberg. Mit der Pinch-Technologie Prozesse und Anlagen optimieren. LfU, 2003.

[62] L. Leible, S. Kälber, G. Kappler, and O. Hurtig. Bereitstellung und Nutzung von SNG. In *5. Kolloquium Sustainable BioEconomy*, 2011.

[63] Linnhoff March. Introduction to Pinch Technology, 1998.

[64] Maximilian Kloess. Wasserstoff und Methan aus erneuerbarer Stromerzeugung - Eine Energiewirtschaftliche Betrachtung, 2013.

[65] T. Melin. *Membranverfahren*. Springer-Verlag, 2007.

[66] T. Melin and R. Rautenbach. *Membranverfahren: Grundlagen der Modul- und Anlagenauslegung*. Springer Verlag, 2003.

[67] Michael Fleige. *Direkte Methanisierung von CO_2 aus dem Rauchgas konventioneller Kraftwerke*. Springer Fachmedien Wiesbaden, 2015.

[68] Michael Meixner. Ermittlung von Auslegungsgrundlagen für die Biomassetrocknung. Master's thesis, TU Wien, 2007.

[69] R. Morand, R. Bendel, R. O. Brunner, and H. Pfenninger. PROZESSINTEGRATION MIT DER PINCHMETHODE, 2006.

[70] M.Saric, J. Dijkstra, S. Walspurger, and W. Haije. THE POTENTIAL OF POWER TO GAS TECHNOLOGY INTEGRATED WITH BIOMETHANE PRODUCTION. Technical report, ECN, 2014.

[71] NETL. Gasification in Detail - Types of Gasifiers - Entrained Flow Gasifiers. Internet, January 2013.

[72] P. Nitschke-Kowsky, J. Schenk, P. Schley, and K. Altfeld. Gasbeschaffenheiten in Deutschland. Gaswärme International, June 2012.

[73] C. U. of Technology. Production of synthetic natural gas from gasified biomass - Process integration aspects. Webpage, May 2016.

[74] H. Pauli. Erfahrungen aus dem Betrieb eines Fernwärmespeichers in Kombination mit Kraft Wärme Kopplung am Beispiel der Linz Strom GmbH. In *Presentation*, 2012.

[75] C. Pfeifer, B. Puchner, and H. Hofbauer. In-Situ CO2-Absorption in a Dual Fluidized Bed Biomass Steam Gasifier to Produce a Hydrogen Rich Syngas. *INTERNATIONAL JOURNAL OF CHEMICAL REACTOR ENGINEERING*, 5:A9, 2007.

[76] T. Pröll. *Potenziale der Wirbelschichtdampfvergasung fester Biomasse - Modellierung und Simulation auf Basis der Betriebserfahrungen am Biomassekraftwerk Güssing*. PhD thesis, Vienna University of Technology, 2004.

[77] T. Pröll, I. G. Siefert, A. Friedl, and H. Hofbauer. Removal of NH3 from Biomass Gasification Producer Gas by Water Condensing in an Organic Solvent Scrubber. *Industrial & Engineering Chemistry*, 44:1576–1584, 2005.

[78] Proton Inc. Product Specification, 2015.

[79] L. Rabou. ESME - ECN System for Methanation. In *Malmö, 30th of October*, 2015.

[80] RAG Rohöl-Aufsuchungs Aktiengesellschaft. Erdgasspeicher. Brochure, June 2014.

[81] B. Rehling. *Assessment of the 1MW BioSNG pilot and demonstration unit on technological and economical aspects*. PhD thesis, Vienna University of Technology, 2012.

[82] B. Rehling, H. Hofbauer, R. Rauch, and C. Aichernig. BioSNG - process simulation and comparison with first results from a 1-MW demonstration plant. *Biomass Conversion and Biorefinery*, 1:111–119, 2011.

[83] Richard van Basshuysen. *Erdgas und erneuerbares Methan für den Fahrzeugantrieb*. Springer Vieweg, 2015.

[84] T. Ringhofer. Evaluierung des Biomassetrockners der Kraft-Wärme-Kopplungsanlage Oberwart. BSc Thesis, 2011. TU Wien.

[85] H. Röder. Marktmodell und Preisprognose für Holzpellets in Deutschland. 14. Industrieforum Pellets. Munich.

[86] P. Rundel, B. Meyer, M. Meiller, I. Meyer, R. Daschner, M. Jakuttis, M. Franke, S. Binder, and A. Hornung. Speicher für die Energiewende. Technical report, Fraunhofer - Institut für Umwelt-, Sicherheits - und Energietechnik UMSICHT Institutsteil Sulzbach - Rosenberg, 2013.

[87] M. Saric, J. Dijkstra, and S. Walspurger. Power-to-gas coupling to biomethane production: a feasibility study. In *International Conference on Polygeneration Strategies*, 2013.

[88] M. Saric, J. Dijkstra, S. Walspurger, and W. Haije. POTENTIAL OF BIOMASS TO GAS CONVERSION CHAIN INTEGRATION WITH BIOMETHANE PRODUCTION. In *ICCE*, 2014.

[89] T. Schildhauer. Treibstoff-Erzeugung mittels Power-to-Gas Verfahren. SSM-Tagung, 2015.

[90] T. Schildhauer and S. Biollaz. HOCHTEMPERATUR-ENTSCHWEFELUNG FÜR BIOGENE PRODUKTGASE - DESIGN UND OPTIMIERUNG. Technical report, Eidgenössisches Departement für Umwelt, Verkehr, Energie und Kommunikation UVEK, 2009.

[91] D. Schollenberger. Nutzung von Wabenreaktoren zur Methanisierung bei PtG-Prozessen. Master's thesis, KIT, 2013.

[92] Serge Biollaz. Stand der Entwicklung im Bereich Holzmethanisierung: Knacknüsse und Fortschritte. Presentation, June 2015.

[93] A. Sharma, V. Tyagi, C. Chen, and D. Buddhi. Review on thermal energy storage with phase change materials and applications. *Renewable and Sustainable Energy Reviews*, 13:31–345, 2009.

[94] SimTech. MDK Documentation.

[95] T. Smolinka, J. Garche, C. Hebling, and O. Ehret. OVERVIEW ON WATER ELECTROLYSIS FOR HYDROGEN PRODUCTION AND STORAGE. In *SYMPOSIUM - Water electrolysis and hydrogen as part of the future Renewable Energy System Copenhagen/Denmark*, 2012.

[96] T. Smolinka, M. Günther, and J. Garche. Stand und entwicklungspotenzial der wasserelektrolyse zur herstellung von wasserstoff aus regenerativen energien. Technical report, Fauenhofer ISE and FCBAT, 2011.

[97] Stefan Rönsch and Andreas Ortwein. Methanisierung von Synthesegasen - Grundlagen und Verfahrensentwicklungen. *Chemie Ingenieur Technik*, 83:1200–1208, 2011.

[98] W.-D. Steinmann and R. Schulte. Thermische energiespeicher zur verstromung diskontinuierlicher abwärme. Technical report, Deutsches Institut für Luft- und Raumfahrt - Institut für technische Thermodynamik, Stuttgart - Stadtwerke Esslingen, 2010.

[99] H. Steinmüller, J. Lindorfer, M. Koppe, P. Biegger, M. Lehner, M. Harasek,

A. Makaruk, M. Miltner, R. Haas, and W. Gawlik. Power to Gas - eine Systemanalyse. Technical report, JKU Linz and MU Leoben and TU Wien, 2014.

[100] M. Sterner. *Bioenergy and renewable power methane in integrated 100% renewable energy systems*. kassel university press GmbH, 2009.

[101] M. Sterner and I. Stadler. *Energiespeicher - Bedarf, Technologien, Integration*. Springer Vieweg, 2014.

[102] M. Stidl. *Prozesssimulation von spezifischen Anwendungsfällen der Zweibett - Wirbelschicht - Dampfvergasungs - Technologie für die Papier - und Zellstoffindustrie*. PhD thesis, Vienna University of Technology, 2012.

[103] K. Strauss. *Kraftwerkstechnik*. Springer, 2009.

[104] Tobias Pröll. *Applied modelling in process engineering and energy technology*. Lecture notes, 2014.

[105] G. Tondl. *Oxyfuel Verbrennung von Klärschlamm*. PhD thesis, TU Wien, 2012.

[106] C. van der Meijden, L. Rabou, A. V. der Drift, B. Vreugdenhil, and R. Smit. LARGE SCALE PRODUCTION OF BIO METHANE FROM WOOD. In *International Gas Union Research Conference IGRC*, 2011.

[107] L. Wang, N. Pereira, and Y. Hung. *Air Pollution Control Engineering*. Springer, 2004.

[108] P. Weiland. Technische anforderungen an die vergärung von energiepflanzen, March 2009. DECHEMA-Fachtagung Bioenergie.

[109] Weindorf and Bünger. Comments on the paper by Baldur Eliasson and Ulf Bossel: The Future of the Hydrogen Economy: Bright or Bleak. www.hyweb.de, 07 2003.

[110] V. Wilk and H. Hofbauer. Analysis of optimization potential in comercial biomass gasifcation plants using process simulation. *Fuel Processing Technology*, 141:138–147, 2016.

[111] Yates and Satterfield. Intrinsic Kinetics of the Fischer-Tropsch Synthesis on a Cobalt Catalyst. *Energy & Fuels*, 5:168–173, 1991.

[112] J. Zarfl. *Methanation of biomass-derived synthesis gas - in situ DRIFTS studies over an alumina supported nickel catalyst* . PhD thesis, ETH Zurich, 2015.

Appendix A - IPSEpro simulation PSE-flowsheets

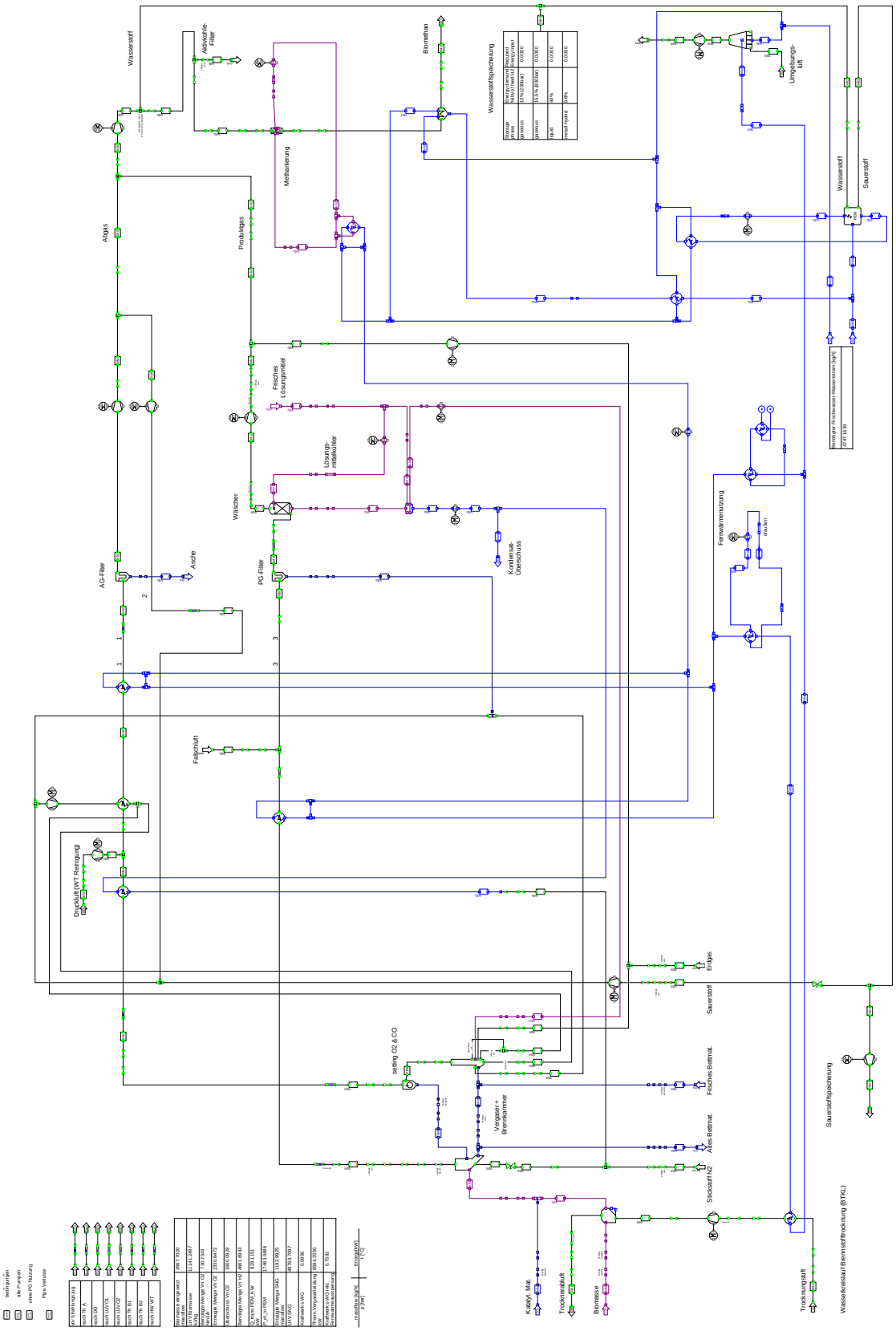


Figure 1: IPSEpro simulation flowsheet of Section 3.1

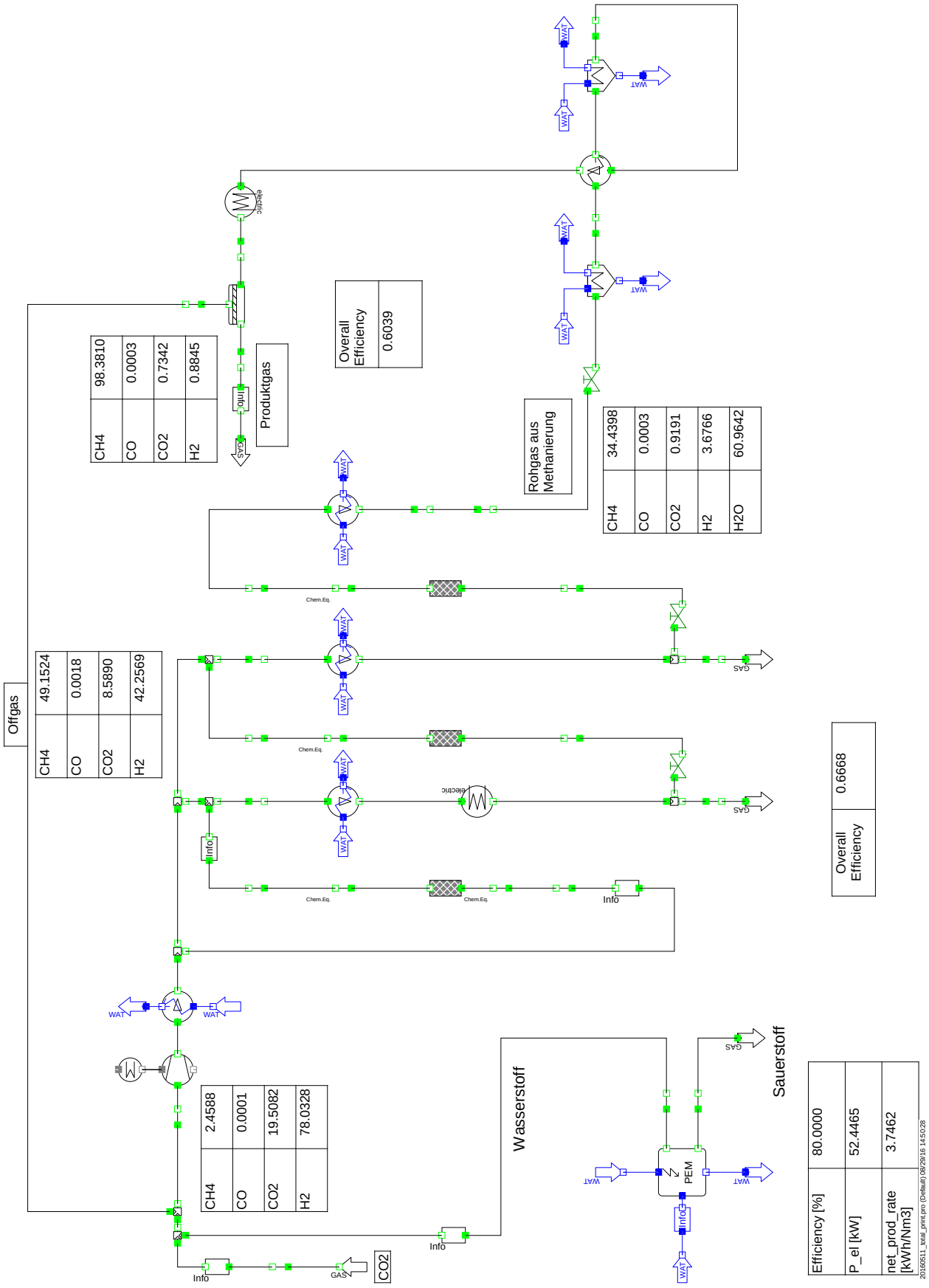


Figure 2: IPSEpro simulation flowsheet of Section 3.2